# REAL TIME CONDITION MONITORING OF ROTORS USING SELF ORGANISING MAPS

By

**Ankur Kumar**

Th
ME/2003/M
K 967

DEPARTMENT OF MECHANICAL ENGINEERING

**Indian Institute of Technology Kanpur**

FEBRUARY, 2003

# REAL TIME CONDITION MONITORING OF ROTORS USING SELF ORGANISING MAPS

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of

**MASTER OF TECHNOLOGY**

February, 2003

*by*

**ANKUR KUMAR**

DEPARTMENT OF MECHANICAL ENGINEERING
**INDIAN INSTITUTE OF TECHNOLOGY**
**KANPUR – 208016 (INDIA)**

# CERTIFICATE

It is certified that the work contained in the thesis entitled **"Real Time Condition Monitoring of Rotors using Self Organising Maps."** by *Ankur Kumar* has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Nalinaksh Vyas.

Dr. Nalinaksh S Vyas
(Professor)
Department of Mechanical Engineering,
Indian Institute of Technology, Kanpur.

February, 2003

## ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and thanks to my thesis supervisor Dr. N.S. Vyas for his guidance, invaluable suggestions and constant encouragement. I am also grateful to him for exposing me to the practical aspects vibration diagnostics. I am also thankful to Dr. V. Raghuram for his valuable suggestions, whenever needed. I am also thankful to DST and persons of GGSS Thermal Power Plant who gave me chance in developing the software for real time condition monitoring of rotors.

I am thankful to Prashant for his invaluable help and tips with Matlab, Rohit, and Nivea for helping me in all possible ways, Prashanth, Himanshu for the lively work environment they have provided.

I appreciate and extend my sincere thanks to Mr. J.P. Verma for his untiring help in sorting out hardware problems. I thank all those who have contributed directly or indirectly to my thesis.

Indian Institute of Technology, Kanpur. Ankur Kumar

February, 2003.

# ABSTRACT

| | |
|---|---|
| Name of student: ANKUR KUMAR | Roll No: Y110503 |
| Degree for which submitted: M.Tech. | Department: Mechanical Engg. |

Thesis title: Real Time Condition Monitoring of Rotors using Self Organising Maps.

Name of thesis supervisor: Dr. N.S.Vyas

Month and year of thesis submission : February 2003

Artificial Neural Networks offer an efficient platform for devising condition monitoring strategies in machinery and plants where the number of components and processes are too many and complex to be mathematically modeled appropriately. This work describes a study on integration of Neural Networks techniques with PC based data acquisition for development of an automated strategy for fault identification on a Machinery Fault Simulator rig. Neural Networks which employ topographic mapping are based on Self Organising Maps. Topographic mapping is commonly detected in many sensory systems. Such mapping serves important information processing functions by relating the physical location of cells on the map surface to specific aspects of information. Unlike commonly used back-propagation and probabilistic neural network training schemes, training for self-organising maps is unsupervised. In the present study, a variety of rotor faults like unbalance, bearing damage, cocked rotor etc., are simulated on the rig in a controlled manner and vibration signals are picked up from multiple stations. Attempt has been made to automate most of the signal processing activities by writing suitable software codes. Data acquisition, time domain display, Fast Fourier Transformation and frequency domain display is done in LabVIEW. Feature extraction from the frequency domain data, and network training codes are written in MATLAB. Training is carried out, for each fault with several sets of vibration data from three channels. Networks based on Back-propagation algorithm, and Probabilistic principles are also studied along with Self-Organising Maps.

## Statement of Thesis Preparation

|  |  |
|---|---|
| Thesis title | "REAL TIME CONDITON MONITORING OF ROTORS USING SELF ORGANISING MAPS " |

|  |  |
|---|---|
| 1. Degree for which submitted | Master of Technology |
| 2. The thesis guide was referred to for thesis preparation | Yes |
| 3. Specifications regarding thesis format have been closely followed | Yes |
| 4. The contents of the thesis were organized according to the guidelines | Yes |

(Signature of Student)
Name: Ankur Kumar
Roll No.: Y110503
Department: Mechanical Engg.

# CONTENTS

# NOMENCLATURE

| | |
|---|---|
| $\Delta_p w_{ji}^h$ | Weight change at layer *h* |
| $\alpha$ | Momentum |
| $\beta$ | Bearing load contact angle |
| $\delta_{pk}^h$ | Individual error for input vector *p* at node *k* in layer *h* |
| $\eta$ , $h_{ci}(t)$ | Learning rate co-efficient |
| $\varphi(.)$ | Activation function |
| $\theta_L^h$ | Threshold for node *L* of layer *h* |
| $\sigma_i$ | Smoothing parameter |
| $D_{ball}$ | Diameter of ball or roller |
| $D_{pitch}$ | Pitch diameter of balls or rollers |
| $E$ | Symptom |
| $E_p$ | Mean square error |
| $F_{bd}$ | Ball Spin defect frequency |
| $F_{cd}$ | Cage defect frequency |
| $F_{ird}$ | Inner race defect frequency |
| $F_{ord}$ | Outer race defect frequency |
| $H_i$ | *i*th fault |
| $K$ | Number of faults |
| $M$ | Number of observations in training symptom pattern *E* |
| $N$ | Number of contained balls or rollers |
| $RPM$ | Rotational speed |
| $f_j^h$ | Activation function at node *j* of hidden layer *h* |
| $n_i$ | Number of training symptom patterns pertaining to *i*th Fault |
| $net_{pj}^h$ | Net input values to *h*th hidden layer |
| $o_{pj}^h$ | Output at output layer units |

# List of Figures

# List of Tables

# CHAPTER 1

# INTRODUCTION

Present day requirements on reliability in power-generation and transportation industries sectors emphasize the importance of efficient Condition Monitoring strategies for Rotating Machinery. Early detection, location and remedy of machinery faults are vital for minimizing the risk of system failure and maintenance costs and maximizing operator safety and efficiency.

Vibration of a machine is a major condition monitoring parameter. Vibration signals hold clues to several types of problems and faults which may occur in a machine. Vibration analysis with appropriate instrumentation can provide reliable knowledge of machine condition. Advances in data acquisition techniques and the developments in the areas of knowledge acquisition, processing and programming make it possible to make the whole process of condition monitoring to be fully automatic which would permit early detection of faults than is possible through conventional limits and trend checks.

Rotating machinery fault diagnosis based on vibration measurements calls for reliable data analysis and decision making in the presence of uncertainty. This uncertainty can be attributed to different factors such as measurement noise, lack of a clear deterministic relationship between the measured quantities and the machine state and variation in the characteristics of the transmission path between the error point and the measurement location due to variations in machine configuration.

There are three phases in the evolution of on-line computer-based vibration monitoring of rotating machinery. The first phase includes collecting the vibration data and other operating parameters that are relevant. Phase two consists of identifying the complicated but normal behavior of the system, accounting for the variation of operating parameters. In phase three, use is made of knowledge-based systems to identify the reasons for departures from normal behavior. Artificial Neural Networks (ANN) find their role in the third phase of a monitoring strategy. Artificial Neural Networks offer an efficient platform for condition monitoring strategies in machinery and plants where the number of components and processes are too many and complex to be mathematically modeled appropriately. Vibration parameters like amplitude, frequency and direction, etc. are widely accepted indicators of the health of a rotating machine.

The present study is an extension of a previous work on detection of faults in rotating machinery using virtual instrumentation (Rao, 2002), who used a Back-Propagation Neural Network Scheme for Condition Monitoring of a Rotor Rig. The focus of the present study is to develop a neural network based on concepts of Self-Organizing Maps, for diagnosing rotor faults. Self Organizing Maps employ Topographic Mapping, which is commonly detected in many sensory systems. Such mapping serves important information processing functions by relating the physical location of cells on the map surface to specific aspects of information. Unlike commonly used back-propagation and probabilistic neural network training schemes, training for self-organising maps is unsupervised. Similar input vectors get grouped in separate clusters on the map. Such maps are used for classification of problems. A variety of rotor faults like unbalance, bearing damage, cocked rotor etc., are simulated on a Machinery Fault Simulator Rig, in a controlled manner and vibration signals are picked up from multiple stations. Attempt has been made to automate most of the signal processing activities by writing suitable software codes. Data acquisition, time domain display, Fast Fourier Transformation and frequency domain display is done in LabVIEW. Feature extraction from the frequency domain data, and network training codes are written in MATLAB. Training is carried out, in each case with several sets of vibration data from four channels. Back-Propagation(BPN), Probabilistic Neural Network(PNN) and Self-Organising Maps(SOM) have been employed.

A brief review of commonly occurring faults in rotating machinery, their symptoms, and fault diagnosis procedures are discussed in Chapter 2. The essential features of Neural Networks with specific reference to Self-Organising Maps have been described in Chapter 3. Chapter 4 describes the experimental set-up, methodology used for data acquisition and the typical fault signatures obtained. Feature extraction, network training and validation are discussed in Chapter 5. Conclusions and scope for future work are listed in Chapter 6.

# CHAPTER 2

# FAULT DIAGNOSIS IN ROTATING MACHINERY

Rotor dynamics has been an area of extensive research during the past two decades. Reference can be made to the studies by Sohre (1991), Rao (1992), Childs (1993) and Dimentberg (1988) for a comprehensive discussion in the field, including fault classification and diagnosis. The most common problems are generally related to *ABC* category Eishleman (1998) – Alignment, Balance and incorrect Clearances (typically on bearings). Prominent rotating machinery problems include – Unbalance, Misalignment, Rolling Element Bearing Defects (Outer Race Defect, Inner Race Defect, Ball Spin Defect, Train/Cage Defect), Cocked Rotor, Rub, Casing Distortion, Asymmetric Shaft, Mechanical Looseness, Oil Film Whirl, Bending Resonance: Gearbox Defects, etc. Both experimental and analytical techniques are routinely used for diagnosis of faults. The most prominent types of rotating machinery problems are discussed below along with the causes and symptoms.

## 2.1 Common Faults in Rotating Machinery

Unbalance

Unbalance is the single most prominent problem of a rotor system that may cause the entire machine to vibrate excessively. This in turn may cause excessive wear in bearings, bushings, shafts, gears, cabling, hoses, cowlings and exhaust systems substantially reducing their service life. When rotation begins, the unbalance exerts centrifugal force tending to vibrate the rotor and its supporting structure. The higher the speed, the greater is the centrifugal force exerted by the unbalance and more violent is the vibration. Centrifugal force increases proportionally to the square of the increase in speed. If the speed doubles, the centrifugal force is quadrupled, etc. Vibration due to unbalance prominently occurs at a frequency equal to the rotor speed. However, specific unbalance forces caused as in the case of ball bearing wear give rise to additional characteristics like high frequency signals corresponding to bearing vibration frequency.

Misalignment

In very broad terms, shaft misalignment occurs when the centerlines of rotation of two (or more) machinery shafts are not in line with each other. In more precise terms, shaft misalignment is the deviation of relative shaft position from a collinear axis of rotation measured at the points of power transmission when equipment is running at normal operating conditions. The misalignment of shafts and other components is one of the most common causes of vibration and failures. Misalignment is present in almost every machine to some extent. Coupling misalignment causes friction and deflection forces, which cause the rotor bearing system to deflect, creating secondary phenomena such as harmonic resonances. Frequencies are characteristically twice the running speed, but first and sometimes third or fourth multiples of running speed are also observed. Misalignment can be classified as parallel or angular.

Rolling Element Bearing Defects

The commonly used of rolling element bearings types are ball bearings, roller bearings, tapered roller bearings, needle bearings. These contain elements that roll between an inner race and an outer race with very close clearances. They provide high stiffness to the rotor but are low in damping. Rolling element bearings have very limited life so it is desirable to monitor these bearings for early indication of impending failure. Application of high loads, shock loads, or bearing attrition due to extended run times increase the internal clearances resulting in bearing failure. Typically four types of defects are observed in rolling element bearings, each giving rise to vibrations at a specific frequency. Each defect excites causes vibrations at different frequencies which are related to the bearing geometry as described next:

| Type of Defect | Frequency | |
|---|---|---|
| Outer Race Defect | $F_{ord} = \left\{\frac{N}{2}\right\} \times \left\{1 + \left(\frac{D_{ball}}{D_{pitch}}\right) \times Cos\beta\right\} \times RPM$ | (2.1) |
| Inner Race Defect | $F_{ird} = \left\{\frac{N}{2}\right\} \times \left\{1 - \left(\frac{D_{ball}}{D_{pitch}}\right) \times Cos\,\beta\right\} \times RPM$ | (2.2) |
| Ball Spin Defect | $F_{bd} = \left(\frac{1}{2}\right) \times \left\{\left(\frac{D_{pitch}}{D_{ball}}\right) - \left(\frac{D_{ball}}{D_{pitch}}\right) \times (Cos\beta)^2\right\} \times RPM$ | (2.3) |
| Train/Cage Defect | $F_{cd} = \left(\frac{1}{2}\right) \times \left\{1 - \left(\frac{D_{ball}}{D_{pitch}}\right) \times Cos\,\beta\right\} \times RPM$ | (2.4) |

where,

| | |
|---|---|
| $N$ | = Number of contained balls or rollers |
| $RPM$ | = Rotational speed of bearing inner race (Hz) |
| $D_{ball}$ | = Diameter of ball or roller |
| $D_{pitch}$ | = Pitch diameter of balls or rollers |
| $\beta$ | = Bearing load contact angle (degrees) |

Cocked Rotor

A rotor with its plane not perpendicular to axis of rotation is termed cocked. Also, unevenness in mass distribution on either side of the plane perpendicular to rotational axis and passing through geometric center of the rotor produces cockiness. This fault generates excessive vibration in axial direction prominently at the first multiple of rotational frequency. Sometimes, second or third multiples of rotational frequency is also observed.

Rotor Rub

The physical contact between rotating elements and stationary machine parts can generate rub condition. Rub can be radial or axial. Mechanical seal rub is a typical example of radial rub, while casing rub at shaft end is a case of axial rub. Mostly both the types of rubs generate vibration at frequency equal to twice the rotational frequency but lower and higher frequencies are also common.

Distortion

Casing and foundation distortion cause vibration in an indirect way, either by generating misalignment between driver and driven machines or by causing internal rub or uneven bearing contact. This in turn transmits forces to the rotor, inducing it to generate forces of its own, such as unbalance and a wide variety of oil film and friction induced forces. Another possibility is that the loads on casing supports shift and can set off a series of resonance problems. Piping forces and foundation distortion often cause this type of difficulty.

Asymmetric Shaft

The response of the asymmetric shaft has several harmonics and the frequencies observed are first, second or third multiples of the rotational frequency and sometimes even higher harmonics, if the asymmetry is predominant.

Mechanical Looseness

Loose rotor components such as disks, sleeves, thrust collars etc. cause internal friction problems. The frequency of vibration is always the rotor critical speed. Mechanical loose components like bolts give rise to $1 \times rev$ and harmonic frequency signals due to secondary phenomena. The amplitude and phase continually change. Loose assembly of bearings give rise to sub-harmonic response and the typical frequency response is $0.5 \times rev$ and $(1/3) \times rev$. This could be mistaken for oil film whirl, particularly in the region of twice the rotor critical speed, around the threshold of instability.

Oil Film Whirl

Oil whirl in an oil lubricated journal bearing can occur due to preload forces, shaft/bearing conditions, shaft eccentricity/concentricity or initial rotor deflection. In such an eventuality a flowing wedge of oil forms in the bearing and drives the shaft ahead of its forward circular motion within the bearing clearance. The frequency of response at the onset of the whirl is always a little less than 0.5*x rev,* in the region of 0.4 to 0.5*x rev.* Generally it occurs around 0.45 to 0.48*x rev.* The frequency gets locked at the critical speed in the unstable region, if the whip is severe. If the bearing is damaged, high frequency response will be observed, due to the rubbing of debris.

Bending Criticals and Resonance

Bending critical of the rotor occurs when the rotational speed is equal to its lateral resonance frequency. Bending critical can be easily detected by synchronous whirl conditions and fairly large amplitude at the rotor speed. Unlike resonance frequency, the critical speed cannot be cured by addition of damping, which in fact may be harmful, causing internal hysteresis. Resonance of the structure, support and auxiliaries cause fairly large amplitudes of vibration at the rotor speed and this occurs over a narrow range of speed of operation. Such resonance can be cured by the addition of damping.

Gearbox Defects

Gearboxes generate additional frequency components equal to running speeds of various shafts associated with them. Also, gear teeth meshing frequencies are prominent. Defects in any of the shafts show frequency components as multiples of the rotational frequency of that shaft. Gear teeth damage/break off show higher amplitudes at gear teeth meshing frequency.

A summary of the rotating machinery defects, characteristics and common remedial measures are given in Table 2.1.

## 2.2 Vibration Analysis As A Predictive Maintenance Tool

Various maintenance techniques can be broadly classified in the following manner as Corrective Maintenance, Preventive Maintenance, Predictive Maintenance. Corrective Maintenance involves replacement of defective parts of a machine once it has been observed to function irregularly. This technique is also commonly referred to as Breakdown Maintenance which means if it broken - fix it! and can only be used for non-critical machinery. With plant downtime becoming more expensive, a failure could be quite expensive in lost production time, and thus preventative maintenance evolved. This involves regular inspections and overhauls at predetermined intervals.

**Table 2.1 Identification and correction of malfunction of rotating machinery**

| Fault | Frequency | Spectrum; Time-domain, Orbit shape | Correction |
|---|---|---|---|
| Mass unbalance | 1X | Distinct 1X with much lower values of 2X, 3X, etc., elliptical and circular orbits | Field or shop balancing |
| Misalignment | 1X, 2X, etc., | Distinct 1X with equal or higher values of 2X, 3X, etc., | Perform hot / cold alignment |
| Shaft bow | 1X | Dropout of vibration around critical speed in Bode plot | Heating or peening to straighten rotor |
| Steam loading | 1X | Load sensitive 1X | Modify admission sequence: repair diaphragms: reinstall nozzles blocks |
| Bearing wear | 1X, sub harmonics, orders | High 1X, high 0.5X, sometimes 1.5 or orders; cannot be balanced | Replace bearing |
| Gravity excitation | 2X | 0.5 critical speed appears on Bode plot | Reduce eccentricity by balancing |
| Cracked rotor | 1X, 2X | High 1X,0.5 critical speeds may show upon coast-down | Remove rotor |
| Looseness | 1X plus large number of orders, 0.5X may show up | High 1X with lower-level orders, large 0.5 order | Shim and tighten bolts to obtain rigidity |
| Coupling lockup | 1X, 2X, 3X .etc., | 1X with high 2X similar to misalignment; may yield different vibration patterns | Replace coupling or remove sludge |
| Thermal instability | 1X | 1X has varying phase angle and amplitude | Compromise balance or remove problem |
| Oil-Whirl | 0.35X-0.47 X | Sub synchronous component less than 0.5 order informal loop in orbit | Temporary: load bearing heavier, correct misalignment; long term: change brg. type |
| Oil-Whip | fn1 | Sub synchronous component does not change with speed | Change bearing type |
| Sub harmonic resonance | 0.5X,1/3X, 0.25X, and higher | Sub synchronous Vibration depending on natural frequency | Remove looseness, excessive flexibility; change natural frequency so it does not match fractional frequencies |
| Hysteresis | 0.65X to 0.85X | High-magnitude fractional frequency(greater than 0.5x) | Eliminate or secure built-up parts |
| Rubs | 0.25X,1/3X, 0.5X | External loops in orbit | Eliminate rub causing conditions as thermal bow and mass unbalance |
| Trapped Fluid | 0.8X to 0.9X | Bearing-sum and different frequencies | Remove fluid in rotor if possible; otherwise eliminate 1X component |

1X=One time operating speed. fn1=First natural frequency

An efficiently planned preventive maintenance program can be quiet effective in discovering the faults in a machine before they reach a terminal state. The main drawback of this method is machines are overhauled while they are still in good condition, and it turns out to be an expensive method. This is also sometimes referred to as PPM (Planned Preventive Maintenance). With the development of condition monitoring techniques, methods were available to determine the actual condition of the machine, and even calculate the time to failure. This enables planners to overhaul a machine when it needed it, and to concentrate on the problem areas. This is known as predictive maintenance. Predictive Maintenance techniques have gained prominence in the last decade.
Some of the benefits from a predictive maintenance program are: (i) early warning of problems allows correction without overtime, with minimal impact to production, and without high expediting costs to procure replacement parts; (ii) elimination of unexpected failures which create extensive, costly repairs and expensive loss of production, whereas early correction would have been inexpensive; (iii) effective scheduling of the maintenance workforce during outages to concentrate on the equipment most in need of repair; (iv) extended maintenance intervals for those machines with healthy vibration signatures; (v) isolating the cause of problems without resorting to shot gunning of the solution; (vi) no capital expenditure or staff involvement is necessary - we do everything required; (vii) no modification to machines or disruption to production.

Various Predictive Maintenance techniques include temperature monitoring, current and voltage monitoring, metallurgical failures analysis, oil analysis, wear debris analysis (e.g. ferrography, atomic absorption, atomic emission, etc.), infrared thermography as well as vibration and sound analysis. In the case of rotating machinery (such as industrial turbines, motors, pumps, fans or gearboxes), Vibration Analysis has become the most popular Predictive Maintenance technique, as the vibration signals measured at critical points on the external surface of a machine (for example, a ball bearing) can contain vital information about the internal processes, and can produce valuable information about the state of development of different possible faults, like machine unbalance, axis misalignment, ball bearing and gear related faults, eccentricities or mechanical looseness.

## 2.3 Diagnostic Techniques

Experimental and analytical techniques, both, have been routinely used for diagnosing faults in operational rotating machinery. These techniques are also used during the design and development processes. Some of these diagnostic techniques are discussed below.

### 2.3.1 Time Domain Or Wave Form Analysis

Time domain signal is an unprocessed record of the vibration as picked up from a transducer. It consists of information about the physical behavior of machine. A serious limitation is that it can be too complex for analysis if excessive noise, signal modification or several frequencies are present. Several magnitude parameters can be extracted from the time history of a noise or vibration signal. They are (i) the peak level (ii) root mean square (rms) level and (iii) the crest factor. Crest factor is the ratio of the peak level to the rms level. It is a measure of the impulsiveness of a noise or vibration signal. In addition to peak levels, rms levels and crest factors, various other statistical parameters can be extracted from the time histories of the noise and vibration signals. These include (i) Probability density distributions, (ii) second, third and fourth order statistical moments. The first two statistical moments of a probability density distribution are the mean value and the mean square value. The third statistical moment is the skewness of a distribution and it is a measure of the symmetry of the probability density function. The fourth statistical moment is widely used in machinery diagnostics, particularly for rolling element bearings. It is called kurtosis. The kurtosis of a signal is very useful for detecting the presence of an impulse within the signal. It is widely used for detecting discrete impulsive faults in rolling element bearings. Good bearings tend to have a kurtosis value ~3, and bearings with impulsive defects tend to have higher values (generally>4). The usage of the kurtosis is limited because, as the damage to a bearing becomes distributed, the impulsive content of the signal decreases with a subsequent decrease in the kurtosis value. However, the time domain data is normally too difficult to analyse in such form as signals are not always periodic and are mostly polluted by noise. Noise can be reduced by signal conditioners but cannot be eliminated altogether. Time between events represents the frequency component specific to the machine.

### 2.3.2 Orbital Analysis

The horizontal and vertical motions of the rotor with respect to a sensor mounted on the bearing are simultaneously displayed on oscilloscope. The orbit represents the instantaneous position of the rotor. The orbit can be used to assess oil whirl and other asynchronous motions as well as synchronous phenomena such as mass unbalance and misalignment. Orbital analysis is also employed to diagnose problems like shaft pre-loading. For example, a horizontal pre-load (e.g. misalignment) restricts horizontal motion and will produce an elliptical orbit with the vertical axis longer than the horizontal. Other defects like torque reaction, pressure angle forces associated with gearboxes, internal seal rub due to casing distortion, shaft rubs etc. can also be studied through orbital analysis.

### 2.3.3 Spectrum Analysis

It is common for information to be encoded in the sinusoids that form a signal. The information on frequency, phase and amplitude of the component sinusoids is obtained from the DFT (Digital Fourier Transform) or the FFT (Fast Fourier Transform). The FFT spectrum of a signal can be obtained standard digital FFT algorithms. Vibration signals of rotating machinery mainly comprise of peaks at the machine's rotating frequency (RF) and its harmonics (2xRF, 3xRF, 4xRF). The magnitude of some of these peaks vary considerably in presence of faults. For example, the spectrum of a machine with mass unbalance will normally show a clear peak at the rotating frequency, while in a misaligned machine the second harmonic of the rotating frequency is usually particularly excited. On the other hand, if a defect appears in one of the ball bearings like for example a small irregularity in the bearing's outer race, inner race, or one of the rolling elements, vibration occurs at frequencies bearing specific relationships with the bearing geometry. Other types of peaks that can be observed in the frequency domain are side bands of the bearing defect frequencies with the rotating frequency. The appearance of side bands in the power spectrum is an effect of the modulation of a signal with frequency $f_1$ by another signal with frequency $f_2$. If, for instance, the periodic signal cos $(2\pi f_1)$ is modulated by the signal cos $(2\pi f_2)$, the resulting signal will be cos $(2\pi f_1)$ $cos$ $(2\pi f_2)$. The power spectrum of such a signal consists of two peaks at frequencies $(f_1 + f_2)$ and $(f_1 - f_2)$ frequencies.

### 2.3.4 Cepstrum Analysis

Cepstrum is the forward Fourier transform of a spectrum. It is thus the spectrum of a spectrum, and has certain properties that make it useful in many types of signal analysis. One of its more powerful attributes is the fact that any periodicities, or repeated patterns, in a spectrum will be sensed as one or two specific components in the cepstrum. If a spectrum contains several sets of sidebands or harmonic series, they can be confusing because of overlap. But in the cepstrum, they will be separated in a way similar to the way the spectrum separates repetitive time patterns in the waveform. Gearboxes and rolling element bearing vibrations lend themselves especially well to cepstrum analysis. The cepstrum is closely related to the auto correlation function. The word cepstrum was coined by reversing the first syllable in the word spectrum. The cepstrum exists in a domain referred to as quefrency (reversal of the first syllable in frequency) which has units of time. The real cepstrum of a digital signal x(n) is defined as:

$$c(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \ln|X(\omega)|\, e^{j\omega n} d\omega \qquad (2.5)$$

and the complex cepstrum is defined as:

$$\hat{x}(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \ln|X(\omega)|\, e^{j\omega n} d\omega \qquad (2.6)$$

where the complex logarithm is used:

$$\hat{x}(\omega) = \ln X(\omega) = \ln|X(\omega)| + j\theta(\omega) \qquad (2.7)$$

$$\theta(\omega) = \arg(X(\omega)) \qquad (2.8)$$

The power cepstrum can be used for the identification of any periodic structure in a power spectrum. It is ideally suited to the detection of periodic effects such as detecting harmonic patterns in machine vibration spectra- e.g. the detection of turbine blade failures, and for detecting and separating different side band families in a spectrum- e.g. gear box faults. Power cepstrum analysis is generally used as a complementary tool to spectral analysis. It helps identify items, which are not readily identified by spectral analysis. Its main limitation is that it tends to suppress information about the over all spectral content of a signal, spectral content, which might contain useful information in it's own right. It is thus recommended that cepstrum analysis always be used in conjunction with spectral analysis. Another type of cepstrum, which is sometimes used in signal analysis, is the complex cepstrum. The complex cepstrum is defined as the inverse Fourier transform of the logarithm of the forward Fourier transform of a time signal.

### 2.3.5 Expert Systems

Conventional manual trend checking methods and diagnosing often require human experts to analyze the vibration signature. In Expert Systems attempt is made to develop an algorithm on the basis of human expertise, which is available. These are often considered to be one of the most popular application of artificial intelligence (AI). Knowledge base is the most important part of any expert system. It is stored in the form of facts and rules. The facts are termed as deep knowledge and rules are simply the heuristics. The knowledge is controlled by an inference engine, which interacts with the user and the knowledge base according to the rules contained in it. Since the knowledge or data in most cases may be incomplete or uncertain, the models employ probabilistic reasoning technique such as Bayes's rule, fuzzy logic, Dempster-Shafer calculus etc. Reference can be made to the work of Nema (1994), who employed probabilistic reasoning techniques for an Expert Shell for diagnosis of rotating machinery.

Table 2.2 gives a summary of the common diagnostic tools for rotating machinery.

**Table 2.2 Diagnostic Techniques for Rotating Machinery**

| Technique | Use | Description | Instrument Type |
|---|---|---|---|
| **Time-domain analysis** | Modulation, Pulses, Phase, Truncation, glitch | Amplitude versus time | Analog and digital oscilloscope, FFT spectrum analyzer |
| **Orbital analysis** | Shaft motion, subsynchronous whirl | Relative displacement of rotor bearing in X,Y directions | Digital vector filter, Oscilloscope |
| **Spectrum analysis** | Direct frequencies, natural frequencies, sidebands, beats, subharmonics, sum and difference frequencies | Amplitude versus frequency | FFT spectrum analyzer |
| **Cepstrum analysis** | Sideband and harmonic frequency measurement, accurate quantification of sideband and harmonic severity. | Inverse Fourier transform of logarithmic power spectrum | FFT spectrum analyzer |

The work carried out in this thesis is based on Artificial Neural Network techniques, which are described in the next chapter.

# CHAPTER 3

# ARTIFICIAL NEURAL NETWORKS

Artificial Neural Networks are also called parallel distributed processing architectures and neuromorphic systems. An artificial neural network (ANN) is an information-processing paradigm inspired by the way the densely interconnected, parallel structure of the mammalian brain processes information. Artificial neural networks are collections of mathematical models that emulate some of the observed properties of biological nervous systems and draw on the analogies of adaptive biological learning. The key element of the ANN paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements that are analogous to neurons and are tied together with weighted connections that are analogous to synapses.

Historically, much of the inspiration for the field of NNs came from the desire to produce artificial systems capable of sophisticated, perhaps "intelligent", computations similar to those that the human brain routinely performs, and thereby possibly to enhance the understanding of the human brain. The exact workings of the human brain are still a mystery. Yet, some aspects of this amazing processor are known. In particular, the most basic element of the human brain is a specific type of cell which, unlike the rest of the body, doesn't appear to regenerate. Because this type of cell is the only part of the body that isn't slowly replaced, it is assumed that these cells are what provide us with our abilities to remember, think, and apply previous experiences to our every action. These cells, all 100 billion of them, are known as neurons. Each of these neurons can connect with up to 200,000 other neurons, although 1,000 to 10,000 is typical. The power of the human mind comes from the sheer numbers of these basic components and the multiple connections between them. It also comes from genetic programming and learning.

The individual neurons are complicated. They convey information to each other via a host of electrochemical pathways. There are over one hundred different classes of eurons, depending on the classification method used. Together these neurons and their connections form a process which is not binary, not stable, and not synchronous. Artificial neural networks try to replicate only the most basic elements of this complicated, versatile, and powerful organism.

Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well. Learning typically occurs by example through training, or exposure to a validated set of input/output data where the training algorithm iteratively adjusts the connection weights (synapses). These connection weights store the knowledge necessary to solve specific problems. Practical applications of NNs most often employ supervised learning. For supervised learning, training data that includes both the input and the desired result (the target value), needs to be provided. After successful training, input data alone can be presented to the NN (that is, input data without the desired result), and the NN computes an output value that approximates the desired result.

Although ANNs have been around since the late 1950's, it wasn't until the mid-1980's that algorithms became sophisticated enough for general applications. Today ANNs are being applied to an increasing number of real- world problems of considerable complexity. They are good pattern recognition engines and robust classifiers, with the ability to generalize in making decisions about imprecise input data. They offer ideal solutions to a variety of classification problems such as speech, character and signal recognition, as well as functional prediction and system modeling where the physical processes are not understood or are highly complex. ANNs may also be applied to control problems, where the input variables are measurements used to drive an output actuator, and the network learns the control function. The advantage of ANNs lies in their resilience against distortions in the input data and their capability of learning. They are often good at solving problems that are too complex for conventional technologies (e.g., problems that do not have an algorithmic solution or for which an algorithmic solution is too complex to be found) and are often well suited to problems that people are good at solving, but for which traditional methods are not.

ANN has been attracting the attention of scientists and technologists for long. McCulloh and Pitts formulated the first formal definition of a synthetic neuron model based on biological model in 1943. However, neural networks being massively parallel-distributed processors require tremendous amount of computation. Non-availability of fast and affordable computing power had limited its growth till the last decade of twentieth century. Application of ANNs to vibration and fault diagnosis are relatively few. Mayes (1994) applied ANN for on-line vibration monitoring of large turbo-generators. The investigations focussed on data processing and the use of neural networks. Elkordy, Chang and Lee (1994) investigated the applicability of ANNs for vibration signature analysis of a five-storey structure. McCormick and Nandi (1997) contemplated the application of ANN for real-time fault classification of rotating shafts. Vyas et. al. (2000) used Sohre's knowledge base to train one each of Back-propagation and Probabilistic Neural Networks to identify rotor faults.

## 3.1 Network Architectures

The manner in which the neurons in a neural network are structured is intimately linked with the learning algorithm used to train the network. Four general classes of network architectures are:

(a) Single Layer Feed Forward Network: This is a network of neurons organized in the form of layers. The simplest form of a network consists of an input layer of source nodes that project on to an output layer of neurons, but not vice versa. This is of feed forward type (Fig. 3.1).

(b) Multi-layer Feed Forward Network: It is similar to the one described in (a) above except that it has one or more hidden layers of neurons. Hidden layers of neurons are interface between the input and output layers. It is observed that some problems converge better when these hidden layers are used (Fig. 3.2).

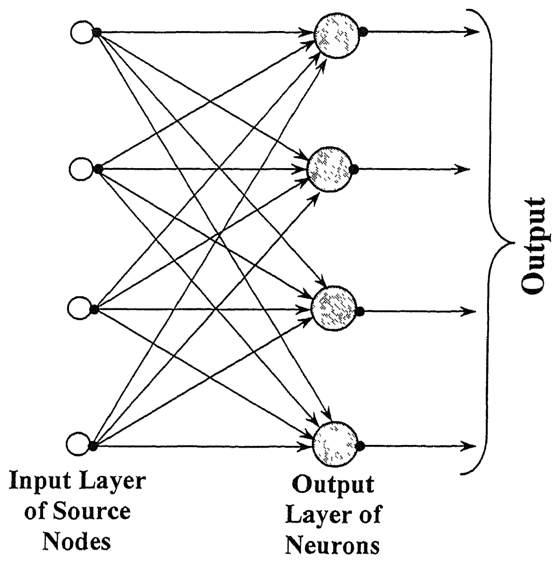


**Fig. 3.1 Single Layer Feed Forward Network**

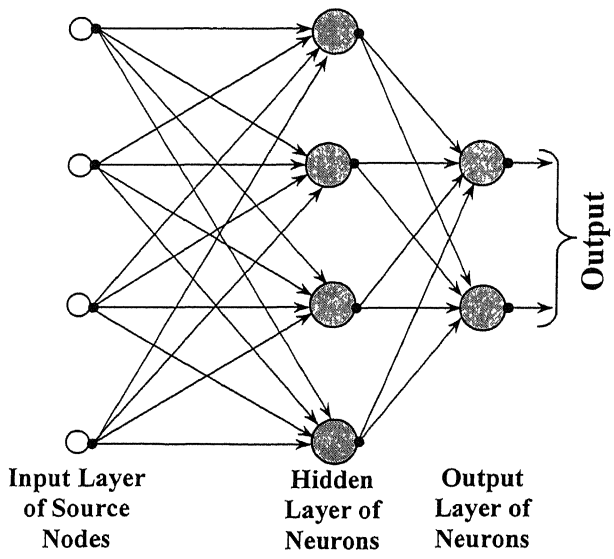


**Fig. 3.2 Feed Forward Network with One Hidden Layer**

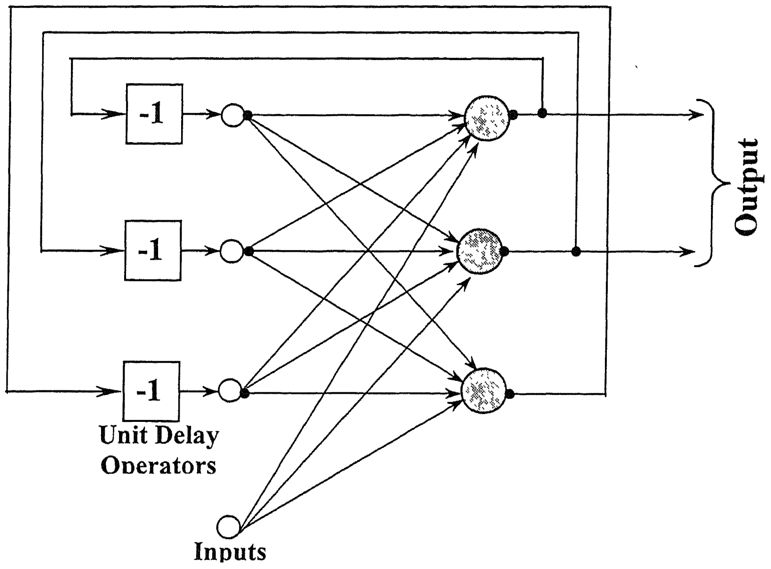


Fig 3.3 Recurrent Network

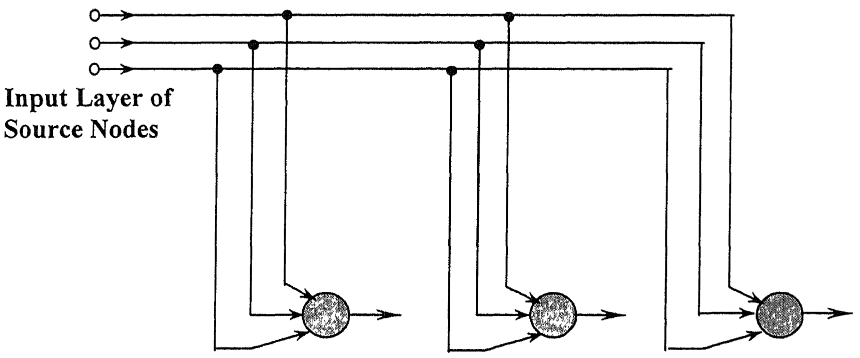


Fig 3.4 One Dimensional Lattice of 3 Neurons

(c) Recurrent Networks: A recurrent network distinguishes itself from the feed-forward network in that it has at least one feedback loop. For example, a recurrent network may consist of single layer of neurons, with each neuron feeding its output signal back to the input of all other neurons (Fig. 3.3). A time delay may be introduced in the feedback path. Recurrent networks typically operate with a discrete representation of data and employ neurons with a hard-limiting activation function.

(d) Lattice Structure: A lattice consists of a one or more multi-dimensional array of neurons with a corresponding set of source nodes that supply the input signals to the arrays (Fig. 3.4). The dimension of the lattice refers to the number of dimensions in which the graph lies.

## 3.2 The Back-propagation Training Algorithm

Since the real uniqueness or 'intelligence' of the network exists in the values of the weights between neurons, we need a method of adjusting the weights to solve a particular problem. For this type of network, the most common learning algorithm is called Back Propagation (BP). One algorithm which has hugely contributed to neural network fame is the back-propagation algorithm. The principal advantages of back-propagation are simplicity and reasonable speed (though there are several modifications which can make it work faster). A BP network learns by example, that is, we must provide a learning set that consists of some input examples and the known-correct output for each case. So, we use these input-output examples to show the network what type of behavior is expected, and the BP algorithm allows the network to adapt. The BP learning process works in small iterative steps: one of the example cases is applied to the network, and the network produces some output based on the current state of it's synaptic weights (initially, the output will be random). This output is compared to the known-good output, and a mean-squared error signal is calculated. The error value is then propagated backwards through the network, and small changes are made to the weights in each layer. The weight changes are calculated to reduce the error signal for the case in question. The whole process is repeated for each of the example cases, then back to the first case again, and so on. The cycle is repeated until the overall error value drops below some pre-determined threshold. At this point we say that the network has learned the problem "well enough" -

the network will never exactly learn the ideal function, but rather it will asymptotically approach the ideal function. Thus back propagation refers to the method for computing the gradient of the case-wise error function with respect to the weights for a feed forward network, a straightforward but elegant application of the chain rule of elementary calculus (Werbos 1974/1994). By extension, back propagation or backprop refers to a training method that uses back propagation to compute the gradient. By further extension, a backprop network is a feed forward network trained by back propagation. "Standard backprop" is a euphemism for the generalized delta rule, the training algorithm that was popularized by Rumelhart, Hinton, and Williams (1986), which remains the most widely used supervised training method for neural nets.

The training scheme for back propagation is supervised training because there is requirement of target vectors for training. The training can be done in batch form (in which the weights are updated after processing the entire training set) and incremental form (in which the weights are updated after processing each case). The algorithm performs the input to output mapping by minimizing a cost function (error function) using a gradient search technique. The cost function, which is equal to the mean squared difference between the desired and the actual net output, is minimized by making wide connection adjustments according to the error between the computed and target output processing element values. When the network starts training, all the weights of the network are first initialized randomly and with these initialized weights, the network output and the difference between the actual and the target output (i.e. the error) is calculated. During the next step, the initialized weights are adjusted to minimize the error by propagating the error backwards. The network outputs and error are calculated again with the updated weights and the process repeats till the error is acceptably small. These two steps are described below and the whole algorithm can be represented schematically as shown in Fig. 3.5.

<u>Forward Pass</u>

The *p*th input vector to the network can be described as

$$x_p = \left(x_{pl} \ldots . x_{pN}\right)' \qquad (3.13)$$

where $x_{pi}$ represents the input attribute *i* for the vector *p*.

The net input to the hidden layer becomes

$$net_{pj}^{h} = \sum_{i=1}^{N} w_{ji}^{h} x_{pi} + \theta_{j}^{h} \tag{3.1}$$

where, $w_{ji}^{h}$ represents the weight of the layer $h$ from node $i$ to node $j$ and $\theta_{L}^{h}$ represents the threshold for the node $L$ of the layer $h$.

The outputs from the hidden layer (which is the input to the output layer) are

$$o_{pj}^{h} = i_{pj} = f_{j}^{h}\left(net_{pj}^{h}\right) \tag{3.2}$$

$o_{pj}^{h}$ is the output from the node $j$ of the hidden layer $h$, $f_{j}^{h}$ is the activation function at node $j$ of the hidden layer $h$.

The net-input values at output layer unit are

$$net_{pk}^{o} = \sum_{j=1}^{L} w_{kj}^{o} i_{pj} + \theta_{k}^{o} \tag{3.3}$$

while the outputs at output units are

$$o_{pk}^{o} = f_{k}^{o}\left(net_{pk}^{o}\right) \tag{3.4}$$

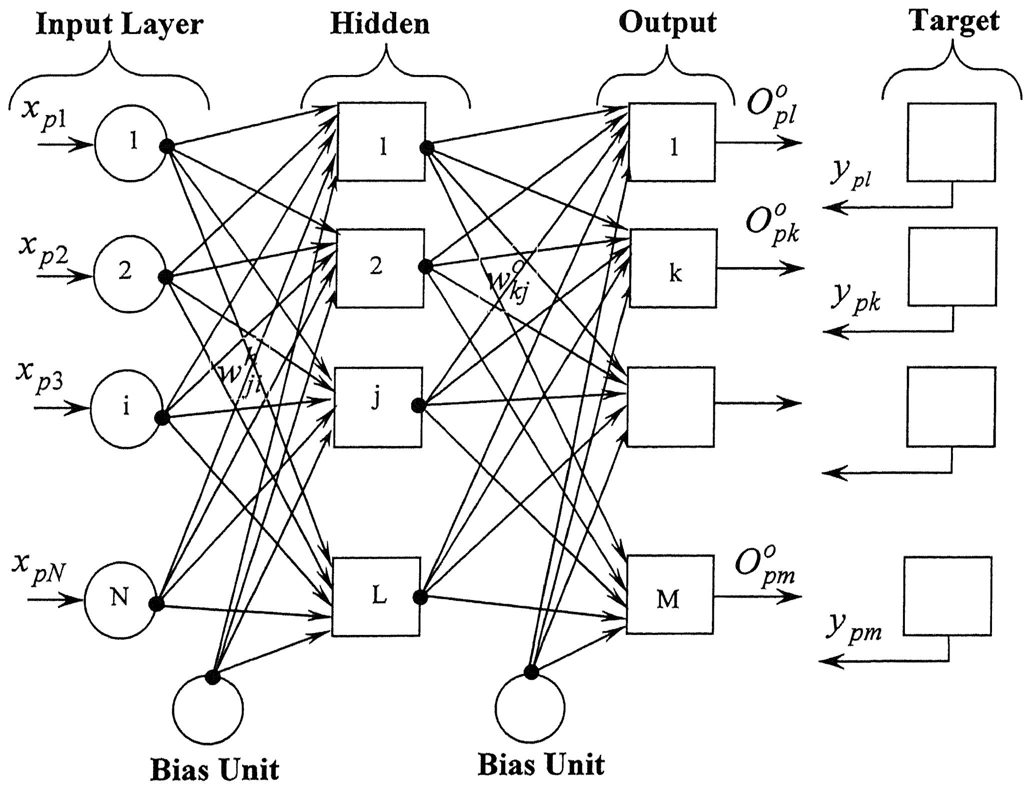


**Figure 3.5 Three Layer Back-propagation Network Architecture**

Individual error at each output unit is

$$\delta_{pk} = y_{pk} - o^o_{pk} \tag{3.5}$$

from which the overall mean square error can be computed as

$$E_p = \tfrac{1}{2}\sum_{k=1}^{m} \delta_{pk}^{\ 2} \tag{3.6}$$

<u>Weights Adjustment Step</u>

Weight adjustment is carried out at the output layer through the following procedure. Using equations (3.5) and (3.6), the mean square error, $E_p$, can be expressed as

$$E_p = \tfrac{1}{2}\Sigma\left(y_{pk} - o^o_{pk}\right)^2 = \tfrac{1}{2}\Sigma\left\{y^o_{pk} - f^o_k\left(net^o_{pk}\right)\right\}^2 \tag{3.7}$$

The weight change of an output layer weight is the negative gradient of $E_p$ with respect to output layer weights $w^o_{kj}$ and can be written as

$$\frac{\partial E_p}{\partial w^o_{kj}} = -\left(y_{pk} - o^o_{pk}\right)\frac{\partial f^o_k}{\partial\left(net^o_{pk}\right)}\frac{\partial\left(net^o_{pk}\right)}{\partial w^o_{kj}} \tag{3.8}$$

However, using equation (3.3)

$$\frac{\partial\left(net^o_{pk}\right)}{\partial w^o_{kj}} = \frac{\partial}{\partial w^o_{kj}}\left(\Sigma\, w^o_{kj} i_{pj} + \theta^o_k\right) = i_{pj} \tag{3.9}$$

and using equation (3.4)

$$\frac{\partial f^o_k}{\partial\left(net^o_{pk}\right)} = f^{o\prime}_k\left(net^o_{pk}\right) = o^o_{pk}\left(1 - o^o_{pk}\right) \tag{3.10}$$

Therefore, the weight change at the output layer weight is

$$\Delta^o_p w = \frac{-\partial E_p}{\partial w^0_{kj}} = \left(y_{pk} - o^o_{pk}\right) f^{o\prime}_k\left(net^o_{pk}\right) \times i_{pj} \tag{3.11}$$

Now denoting

$$\delta^o_{pk} = \left(y_{pk} - o^o_{pk}\right) f^{o\prime}_k\left(net^o_{pk}\right) \tag{3.12}$$

the weight change at the output layer weights can be written as

$$\Delta^o_p w = \delta^o_{pk} i_{pj} \tag{3.13}$$

To make the learning process smooth and to ensure that the weight changes take place in the same direction, two network parameters – learning rate coefficient $\eta$ and momentum $\alpha$. – are introduced in lieu of direct application of the above mentioned weights, so that

$$w_{kj}^{o}(t+1) = w_{kj}^{o}(t) + \eta\left(y_{pk} - o_{pk}^{o}\right) f_{k}^{o'}\left(net_{pk}^{o}\right) \times i_{pj} + \alpha w_{kj}^{o}(t-1) \tag{3.14}$$

A small value of $\eta$ implies, the network has to go through a large number of iterations. Its value is normally kept between 0.05 and 0.9. It is often possible to increase its value as the network error decreases, thereby increasing the speed of convergence. Another way to increase convergence speed is by adopting an extra momentum term while updating the weights. This additional term tends to keep the weight changes in the same direction.

While updating the weights for the hidden layers it should be noted that there is no target output and therefore the adjustment of weights is proportional to their initial contribution.

From equations (3.3) and (3.4) one gets

$$E_p = \frac{1}{2}\sum_{p}\left\{y_{pk} - f_k^{o}\left(\sum w_{kj}^{o} i_{pj} + \theta_k^{o}\right)\right\}^2 \tag{3.15}$$

The weight change of hidden layer weights is the negative gradient of $E_p$ with respect to hidden layer weights $w_{ji}^{h}$ and is given by

$$\frac{\partial E_p}{\partial w_{ji}^{h}} = -\sum_{k}\left(y_{pk} - o_{pk}^{o}\right)\frac{\partial o_{pk}^{o}}{\partial\left(net_{pk}^{o}\right)}\frac{\partial\left(net_{pk}^{o}\right)}{\partial i_{pj}}\frac{\partial i_{pj}}{\partial\left(net_{pj}^{h}\right)}\frac{\partial\left(net_{pj}^{h}\right)}{\partial w_{ji}^{h}} \tag{3.16}$$

and the individual terms on the right hand side of the above equation can be expanded as

$$\frac{\partial o_{pk}^{o}}{\partial\left(net_{pk}^{o}\right)} = \frac{\partial\left(f_k^{o}\left(net_{pk}^{o}\right)\right)}{\partial\left(net_{pk}^{o}\right)} = f_k^{o'}\left(net_{pk}^{o}\right) = o_{pk}^{o}\left(1 - o_{pk}^{o}\right) \tag{3.17}$$

$$\frac{\partial\left(net_{pk}^{o}\right)}{\partial i_{pj}} = \frac{\partial\left(\sum_{j=1}^{L} w_{kj}^{o} i_{pj} + \theta_k^{o}\right)}{\partial i_{pj}} = w_{kj}^{o} \tag{3.18}$$

$$\frac{\partial i_{pj}}{\partial(net^h_{pj})} = \frac{\partial\left(f^h_j net^h_{pj}\right)}{\partial(net^h_{pj})} = f^{h'}_j(net^h_{pj}) = o^h_{pj}\left(1 - o^h_{pj}\right) \tag{3.19}$$

$$\frac{\partial\left(net^h_{pj}\right)}{\partial w^h_{ji}} = \frac{\partial\left(\sum_{i=1}^{N} w^h_{ji} x_{pi} + \theta^h_j\right)}{\partial w^h_{ji}} = x_{pi} \tag{3.20}$$

Substituting equations (3.17)-(3.20) in equation (3.16) one gets

$$\Delta_p w^h_{ji} = -\frac{\partial E_p}{\partial W^h_{ji}} = x_{pi} o^h_{pj}\left(1 - o^h_{pj}\right)\sum_k \left(y_{pk} - o^o_{pk}\right) o^o_{pk}\left(1 - o^o_{pk}\right) w^o_{kj} \tag{3.21}$$

Network parameters $\eta$ and $\alpha$ can be introduced in a manner similar to that in the case of the output layer, to express the final weight change at the hidden layer as

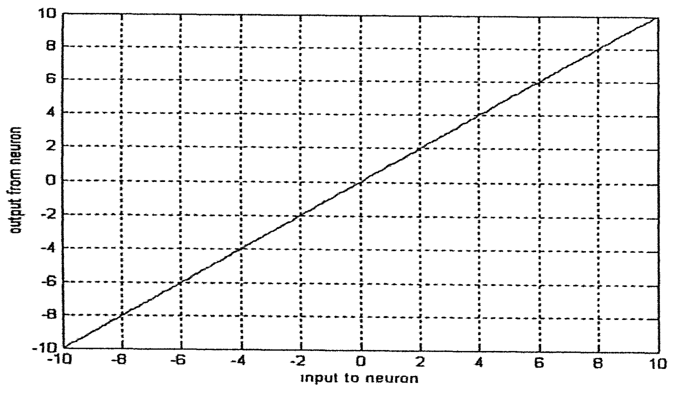
$$w^h_{ji}(t+1) = w^h_{ji}(t) + \eta x_{pi} o^h_{pj}\left(1 - o^h_{pj}\right)\sum \left(y_{pk} - o^o_{pk}\right) o^o_{pk}\left(1 - o^o_{pk}\right) w^o_{kj} + \alpha w^h_{ji}(t-1) \tag{3.22}$$

<u>Activation Functions</u>

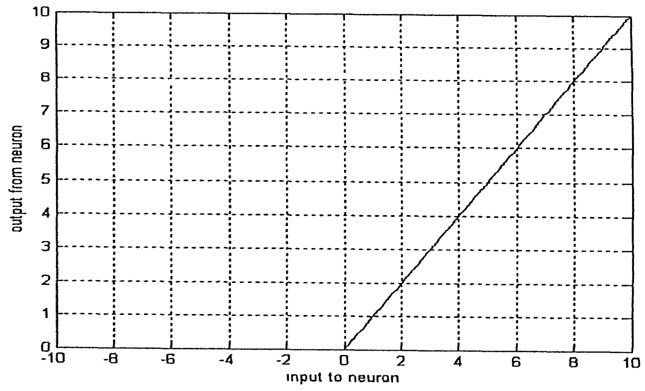
Activation functions for the hidden units are needed to introduce non-linearity into the network. Without non-linearity, hidden units would not make nets more powerful than just plain perceptrons (which do not have any hidden units, just input and output units). Almost any nonlinear function does the job, although for back propagation learning it must be differentiable and it helps if the function is bounded; the sigmoidal functions such as logistic and tanh and the Gaussian function are the most common choices. Some of the common transfer functions are described below.

| <u>Name</u> | <u>Mathematical Representation</u> |
|---|---|
| Linear Activation Function | $\varphi(x) = x$ |
| Positive Linear Activation Function | $\varphi(x) = x$ for $x \geq 0$, <br> $= 0$ for $x \leq 0$. |
| Hyperbolic Linear Tangent Sigmoid | $\varphi(x) = \dfrac{2}{1 + \exp(-\rho x)} - 1$ |
| Logistic Sigmoid Activation Function | $\varphi(x) = \dfrac{1}{1 + \exp(-\rho x)}$ |

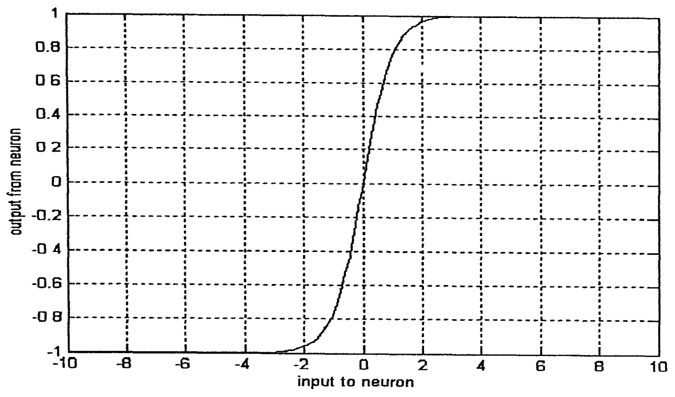
where $\rho$ is a constant, and $\varphi(.)$ is the activation function. The above four activation functions are represented graphically in Fig. 3.6.



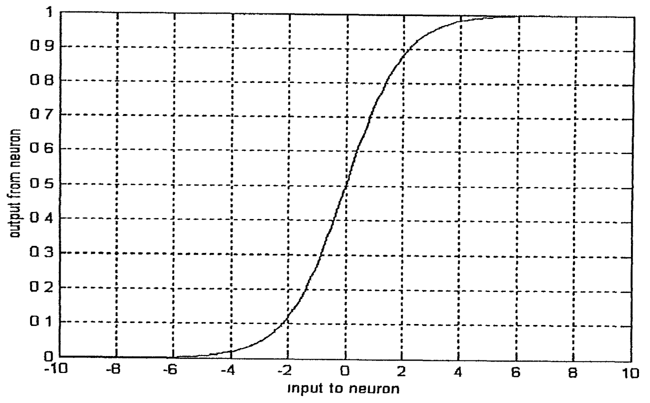
(a) Positive Linear



(b) Pure Linear



(c) Hyperbolic Tangent Sigmoid



(d) Logistic Sigmoid

**Figure 3.6 Activation Functions**

## 3.3 Probabilistic Neural Network (PNN)

Probabilistic Neural Networks (PNNs) are reformulations of Bayes-Parzen classification method. Their training takes into account the probabilities of category membership. An attractive feature is that they deal with outliers better than other statistical and neural network methods. The PNN has four layers of nodes:

- Input layer, comparable to that of the multilayer perceptron, serves to distribute input variables to the network.
- First hidden layer, also known as the pattern layer, contains one node for every training example.
- Second hidden layer, also known as the summation layer, contains one node for each category or class.
- Output layer, a single node acting as a threshold discriminator, identifies the maximum input from the summation nodes.

Thus the probabilistic neural network, in addition to the input layer, has two hidden layers and an output layer (Fig.3.7).

Investigations were also carried out, during the present study on the suitability of probabilistic neural networks for fault identification. Probabilistic Neural Networks (PNNs) find their application mainly in classification problems (Bose and Liang, 1996). Some of the salient features of such networks are briefly described here.

A Probabilistic Neural Network bases itself on Bayes' Rule, which describes the probability of the presence of a particular fault conditional to the observation of a certain symptom as

$$P(H_i / E) = \frac{P(E / H_i)\, P(H_i)}{\sum_{n=1}^{k} P(E / H_n) P(H_n)} \tag{3.23}$$

where

| | |
|---|---|
| $P(H_i / E)$ | the probability that fault $H_i$ is present, given a symptom $E$ |
| $P(E / H_i)$ | probability that symptom $E$ will be observed when fault $H_i$ is present in absence of any specific symptom |

$P\left(H_i\right)$ the a-priori probability that fault $H_i$ is present in absence of any specific symptom.

$k$ number of faults

The following computational procedure can be followed.

$P\left(H_i\right)$ Information generally obtained from machine history, or otherwise taken as unity for all possible faults

$\sum_{n=1}^{k} P\left(E/H_n\right)P(H_n)$ This term in the denominator is constant for all $i$

Given a symptom $E$, the network computes the conditional probability $P\left(H_i/E\right)$, of the presence of faults ($i$ = 1,2 ... $k$). A comparison is then made:

if $P\left(H_i/E\right) > P\left(H_j/E\right)$ for all $j \neq i$, then the fault is $H_i$, or

if $P\left(E/H_i\right) > P\left(E/H_j\right)$ for all $j \neq i$ then the fault is $H_i$.

$P\left(E/H_i\right)$ is computed using the following relationship

$$P(E/H_i) = \frac{1}{(2\pi)^{m/2}\sigma_i^{\,m} n_i} \sum_{j=1}^{n_i} \left[\frac{-(E - E_j^i)^T (E - E_j^i)}{2\sigma_i^2}\right] \tag{3.24}$$

where

$m$ number of observations $e$ in every training symptom pattern $E$

$n_i$ number of training symptom patterns pertaining to the $i$th fault $H_i$

$\sigma_i$ smoothing parameter computed using radial basis functions

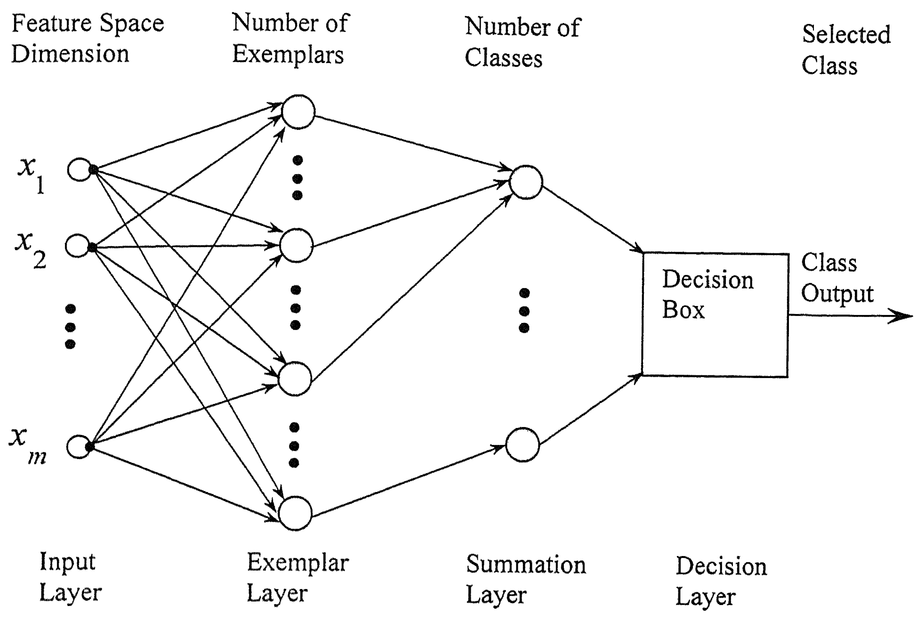


**Fig. 3.7 Probabilistic Neural Network**

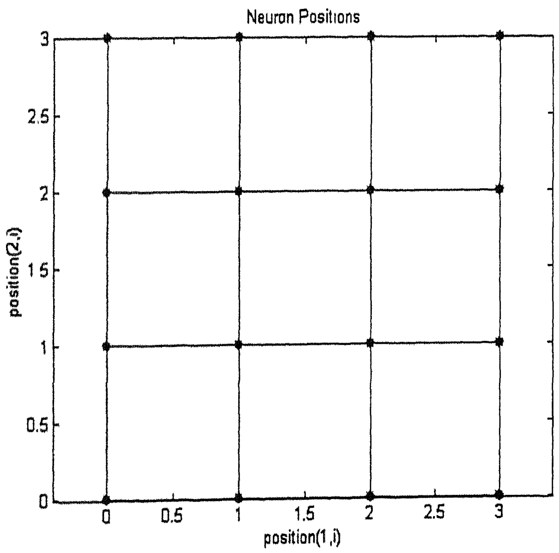
Its major difference with a back-propagation network is that it can be constructed after only a single pass of the training data sets. Also, the activation function is statistically derived from estimates of the probability density functions based on training patterns.

## 3.4 Self-Organising Maps

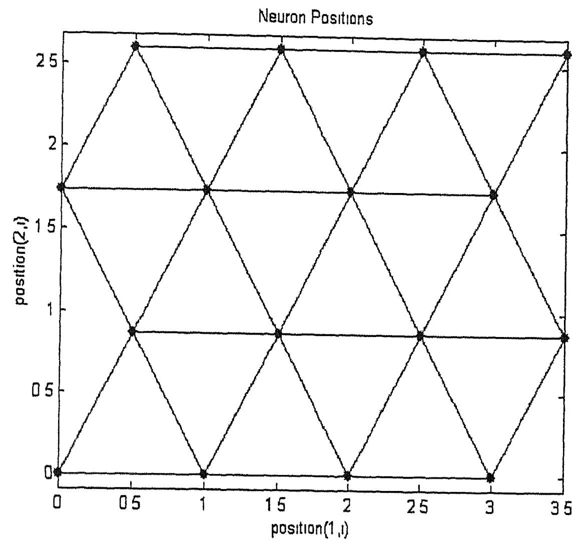
Self-Organizing Map (SOM) is an interesting artificial neural network algorithm. It is a method for producing ordered low dimensional representation of an input data space. These maps can successfully approximate a high-dimensional input space by extracting invariant features of the input signals and maintaining topological relationship between them in lower dimensions. In a topology-preserving map, units located physically next to each other will respond to classes of input vectors that are likewise next to each other. The self-organizing map algorithm can therefore be most usefully applied in the real time monitoring of complex stochastic signals.

The basic Self-Organizing Map (SOM) can be visualized as a sheet-like neural-network array, the cells (or nodes) of which become specifically tuned to various input signal patterns or classes of patterns in an orderly fashion. The learning process is competitive and unsupervised, meaning that no teacher is needed to define the correct output (or actually the cell into which the input is mapped) for an input. In the basic version, only one map node (winner) at a time is activated corresponding to each input. The locations of the responses in the array tend to become ordered in the learning process as if some meaningful nonlinear coordinate system for the different input features were being created over the network.

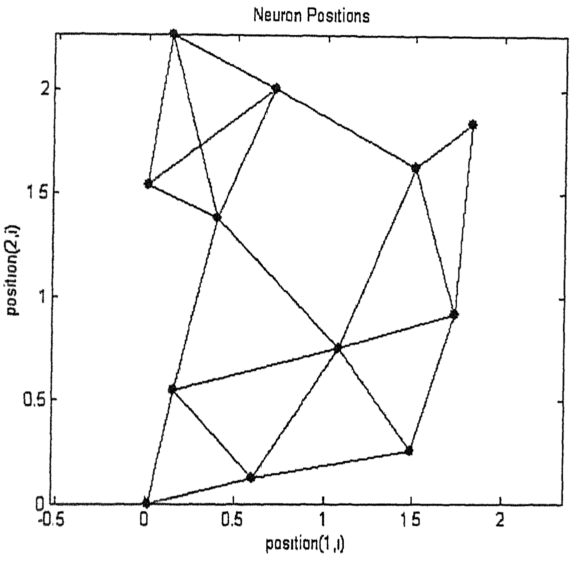
Self organizing learn to detect regularities and correlations in their input and adapt their future responses to that input accordingly [Kohonen, T. Self Organisation and ssociative Memory, 2nd Edition, Berlin, Springer-Verlag, 1987.,Neural Network Toolbox, 1998, The Mathworks Inc., Natick, MA]. The neurons learn to recognize groups of similar inputs in a competitive manner. Neurons in close neighborhood tend to respond to a particular input class while those at a different topographical location respond to another class of inputs. Such a topographic representation of neurons seems to emulate some aspects of information processing of the human brain. The output space is usually a set of units arranged either in a one dimensional line or a two dimensional plane. The output space can have more than two dimensions also. Figs. 3.8 (a), (b) and (c) show three common two-dimensional grids of neurons (rectangular, hexagonal and random).



**(a) Square- Grid**



**(b) exagonal grid**



**(c) Random Grid**

**Fig. 3.8 Neuronal Grids**

Each neuron in the grid is assigned with a weight, depending on its location. On application of the input vector to the network, its distances with the various neurons on the grid are computed. The distances can be Euclidian distances between the two vectors or can be defined in other convenient ways. The process is represented in Fig. 3.9.

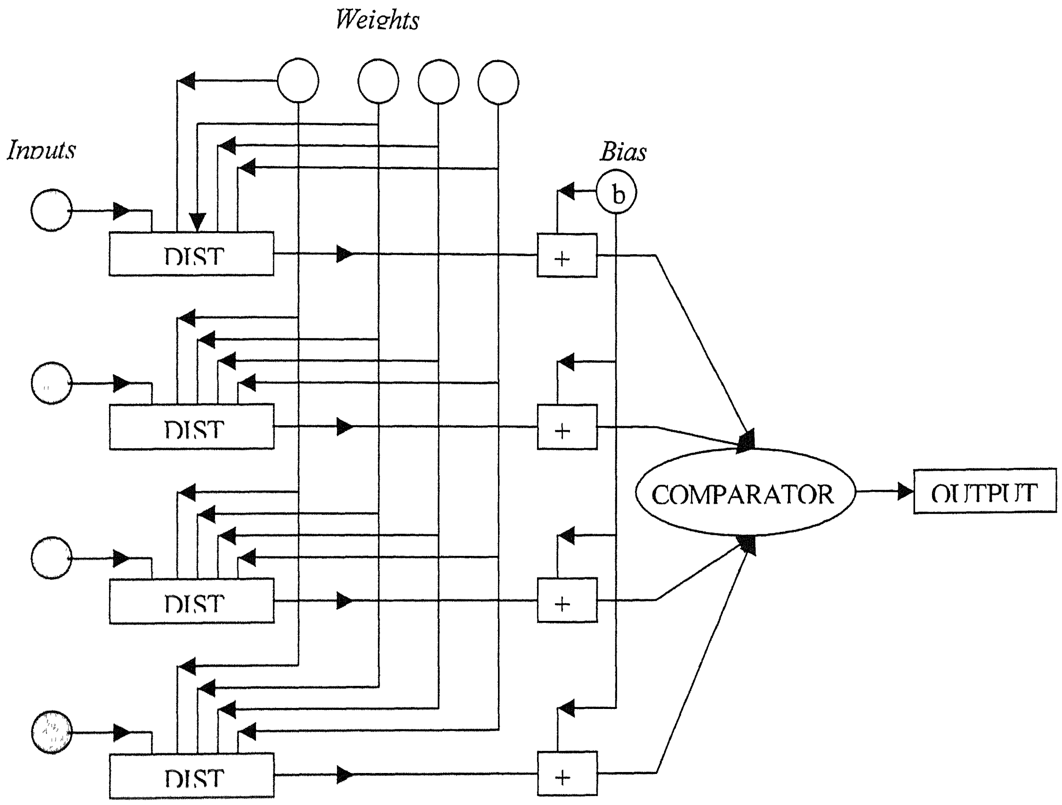


**Fig.3.9 Network Architecture**

The distances are compared and the neuron closest to the input is identified as winner and its weights for the next pass of inputs are modified according to the Kohonen learning rule

$$w(i+1) = w(i) + h_{ci}(t)(x(i+1) - w(i)) \tag{3.25}$$

where $w$ is the weight vector , $x$ is the input vector and $h_{ci}(t)$ is learning rate.

Thus the neuron whose weight vector is closest to the input vector is brought even closer, increasing the possibility of the winning neuron, winning the next time still higher. In addition to updating the weights of the winning neuron, all neurons, within its certain neighbourhood $N$, can be adjusted. Thus when an input vector is presented to the network, the winning neuron and its neigbours move closer to it.

Thus the training algorithm can be summarized as below:

1. For each node $i$ set the initial weight vector $w_i(0)$ randomly.

2. Set the initial neighbourhood $N_i(0)$ to be large (in terms of layers of neurons).
3. Give input - $x(t)$, vector $x$ at a time $t$ ($0 < t < n$, where $n$ is the no. of iteration defined by the user) to all nodes in the network simultaneously.
3. Calculate the winning node.

   Calculate node $c$ with smallest distance between weight vector and input vector

   $$\| x(t) - w_c(t) \| = \min_i \{\| x(t) - w_i(t) \|\} \tag{3.26}$$

   hence

   $$c = \arg\min_i \{\| x(t) - w_i(t) \|\} \tag{3.27}$$

4. Update weights for c and node those within neighborhood

   $$\begin{aligned} w_i(t+1) &= w_i(t) + h_{ci}(t)[x(t) - w_i(t)] \qquad &\text{if } i \in N_i(t) \\ &= w_i(t) &\text{otherwise} \end{aligned} \tag{3.28}$$

5. Present next input of the same fault, after reducing the learning rate $h_{ci}(t)$ so that $h_{ci}(t+1) < h_{ci}(t)$ and the neighborhood set so that $N_i(t+1) \cap N_i(t)$ for $i$.

   Repeat from step 2 choosing a new unique input vector $x(t+1) \neq x(j), j \leq t$ until all iterations have been made.

Fault simulation on the rotor rig, data acquisition and processing and network training and validation are described next in Chapters 4 and 5.

# CHAPTER 4

# EXPERIMENTAL SET-UP AND DATA ACQUISITION

Experimentation is carried out on a specially designed rig called the Machinery Fault Simulator (MFS). The MFS is the comprehensive machine for simulating, studying and learning vibration signatures of machinery defects. The rig is connected to data acquisition system through proper instrumentation. Accelerometers and proximity pick-ups are used for picking up the vibration signals from various stations. Data is acquired and processed by Virtual Instrumentation techniques of LabVIEW software. After collecting the vibration signals and extracting features the vector is fed to the neural network to train it. A trained Neural Network is embedded into the LabVIEW software. This integrates the entire process of acquiring the data, processing of data and testing for fault diagnosis into a single package.

## 4.1 Experimental Test Rig

The Machinery Fault Simulator (make Spectraquest, type 2) is a desktop rotor model which consists of a shaft supported in two roller bearings and driven by a DC motor (Figs. 4.1 and 4.3). A flexible coupling is used to compensate for any misalignment between the bearings and/or motor, as well as preventing any unwanted dynamic effects occurring in the motor being transmitted through the coupling into the rotor itself. Discs can be mounted on the shaft at desired locations. There is provision for connecting a gearbox through a belt drive consisting of two V-belts. The kit also carries a reciprocating mechanism, which can in turn be connected through the gearbox. The rig has an adjustable speed range of 0 - 6000 rpm (0-100 Hz). Different loading conditions can be simulated in MFS by setting torque in an adjustable torque magnetic brake.

## 4.2 Instrumentation

During the present study, six sensors were used to pick up the vibration signals at the following locations (Figs. 4.2 and 4.3):

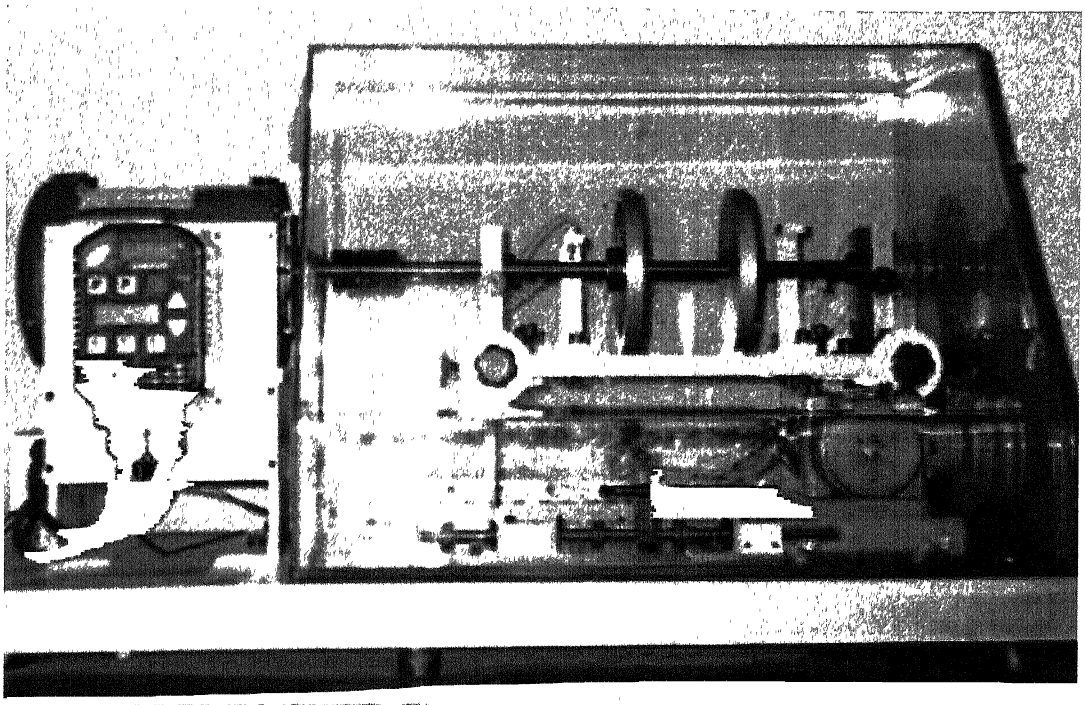


**Fig 4.1 Fault Simulator (make SpectraQuest)**

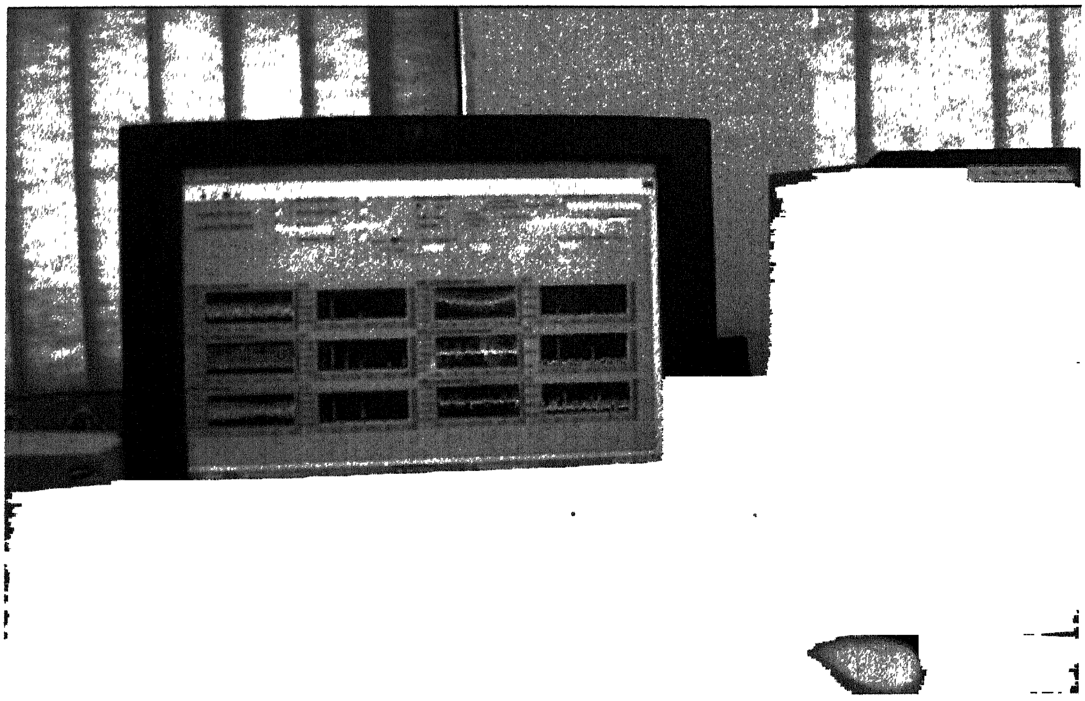


**Fig 4.2 Instrumentation and Data Acquisition Devices**

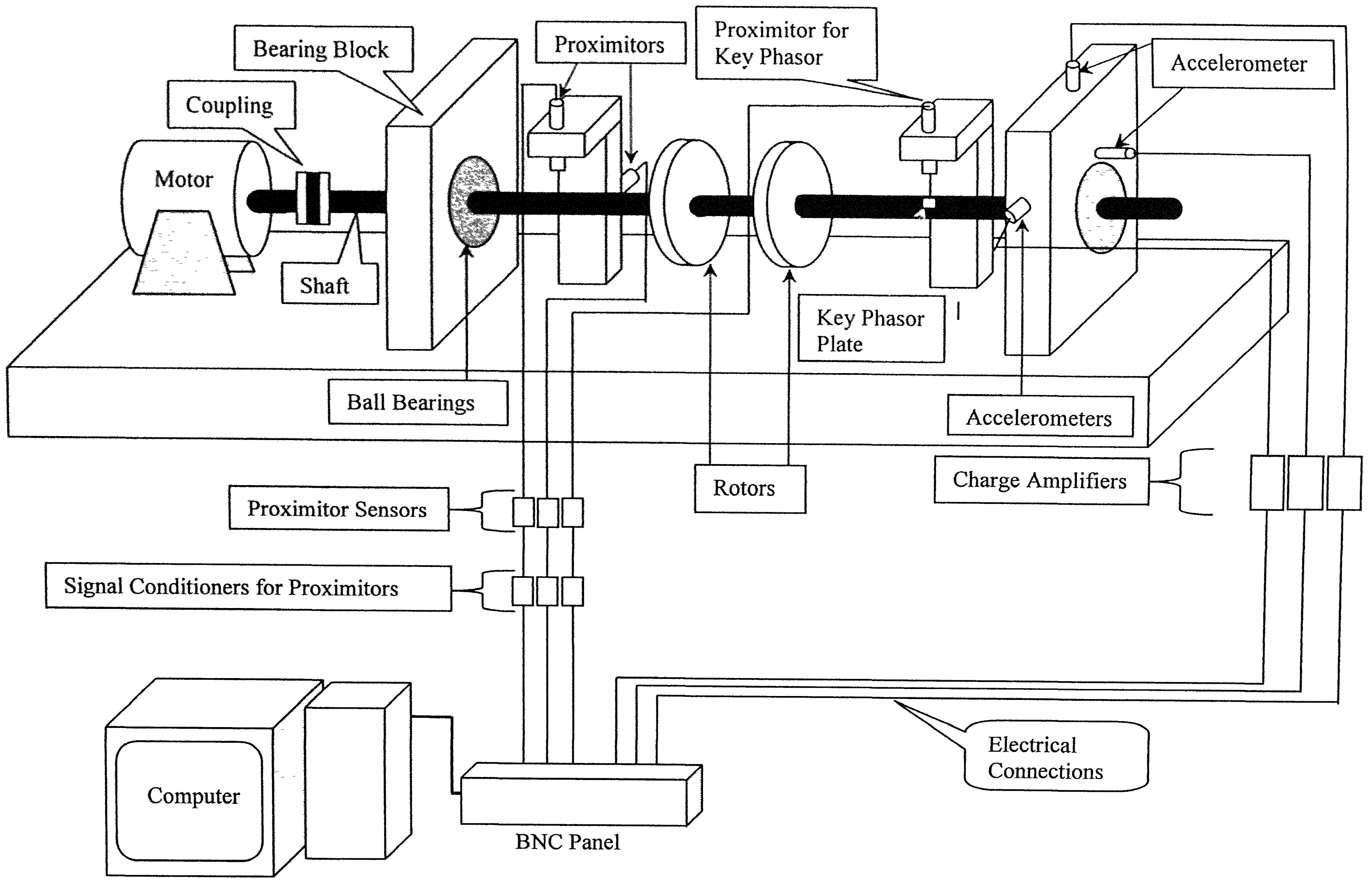


**Figure 4.3 Schematic Diagram of the Experimental Set-up and Instrumentation**

(i) Shaft displacement in the vertical direction at left side bearing end: Sensor - non-contact type eddy current displacement probes (Proximitor), model 3300 RAM, Bentley Nevada make; associated transducer and TK 15 signal conditioner.

(ii) Shaft displacement in the horizontal direction at left side bearing end: Sensor - same configuration as in 1 above.

(iii) Key phasor measurement for phase information (a small plate externally attached to the shaft was used to produce the required trigger signal): Sensor - same configuration as in 1 above.

(iv) Bearing housing vibration, in the vertical direction at the right side bearing: Sensor - accelerometer (model 4374, Bruel and Kjaer make) along with a charge amplifier, (model 2635, of the same make).

(v) Bearing housing vibration, in the Horizontal direction at the right side bearing: Sensor - same configuration as in 4 above.

(vi) Bearing housing vibration, in the axial direction at the right side bearing: Sensor - same configuration as in 4 above.

Vibration data is obtained simultaneously from the above six stations and is acquired into a P-4 computer using a Data Acquisition Card (PCI-6024E, National Instruments, Texas. 200 K sample per second, 12 bit, 16 analog inputs) and LabVIEW software. LabVIEW is a graphical programming language, unlike C or BASIC which are text base language. The program of LabVIEW is called virtual instrument because its appearance, and its operation that can imitate actual instruments. On the front panel of the program you will see the graphs icons knobs etc like actual instruments but one can controls them programmatically. LabVIEW is a data flow programming language in which any node will execute only when it will have its all inputs. It is a general-purpose programming system and includes various tools for data acquisition, data analysis, data presentation, and storage. An interactive user interface of a VI, called *front panel* is used to pass on user-defined parameters into the program, called *block diagram* and program's output parameters are displayed in the front panel. Since user interactive programs can be written for various applications and output parameters of the program can be displayed in text as well as graphical form, it offers tremendous potential for developing wide variety of applications. The front panel is used for data input in form of text box, knobs, buttons etc. and output in the form of text box, figure box, graphs, plots etc. Modular programming techniques make LabVIEW suitable for simulating a large number of instruments limited only by the data acquisition card used.

## 4.3 Data Acquisition, Storage and Display

For data acquisition, data storage and display virtual instrument is made using LabVIEW. This program has following features:

(a) Time domain vibration data acquisition and display from six channels with following user controls (figures/text within brackets indicate default values):

  (i) Scan Rate (2621).

  (ii) Number of data points to be read before display (4096).

  (iii) Device and Channel numbers from which to acquire data (Device No. 1 and Channel Nos. 0 to 5).

(b) A Hanning window is then applied to the acquired time domain signal. Fast Fourier Transform (FFT) of the time signal is carried out and the FFT is displayed in 'real' time along with the time domain signal on the front panel. Provision is also given on the front panel for the user to select the desired frequency range for FFT display.

(c) Option for logging the time domain data and frequency domain data into the hard disc at any desired instant of time.

(d) Control for storing the average FFT data for desired number .

(e) *Run Neural Network and Diag*nose control for on line testing of the faults in the rotor.

(f) The *output of the neural network*, which is indicative of the *fault* predicted. Circular Indicators, one for each type of fault, are provided on the Front Panel, which are *GREEN* in colour in absence of any fault. If a particular fault occurs the indicator pertaining to the fault turns *RED*. An *AUDIO ALARM* is also sounded simultaneously to warn the operator.

Figs. 4.4 (a) shows the Front Panel of the VI. The Block Diagram of the VI is given in Fig. 4.4 (b).

## 4.4 Fault Simulation

Different separate and combined faults were introduced on the Machinery Fault simulator. The list of various faults which were introduced for this study is given below:

(i) No Fault – same as factory condition of MFS

(ii) Mass unbalance on right rotor

(iii) Parallel Misalignment

(iv) Angular Misalignment

(v) Damaged inner race defect on right rotor

(vi) Damaged outer race defect on right rotor

(vii) Ball spin defect on right bearing

(viii) Cocked left rotor

(ix) Loose Belt with Gear

(x) Tight Belt with Gear

(xi) Reciprocating

(xii) Parallel Misalignment +Unbalance

(xiii) Angular Misalignment +Unbalance

(xiv) Bearing Inner Race Fault + Unbalance

(xv) Bearing Outer Race Fault + Unbalance

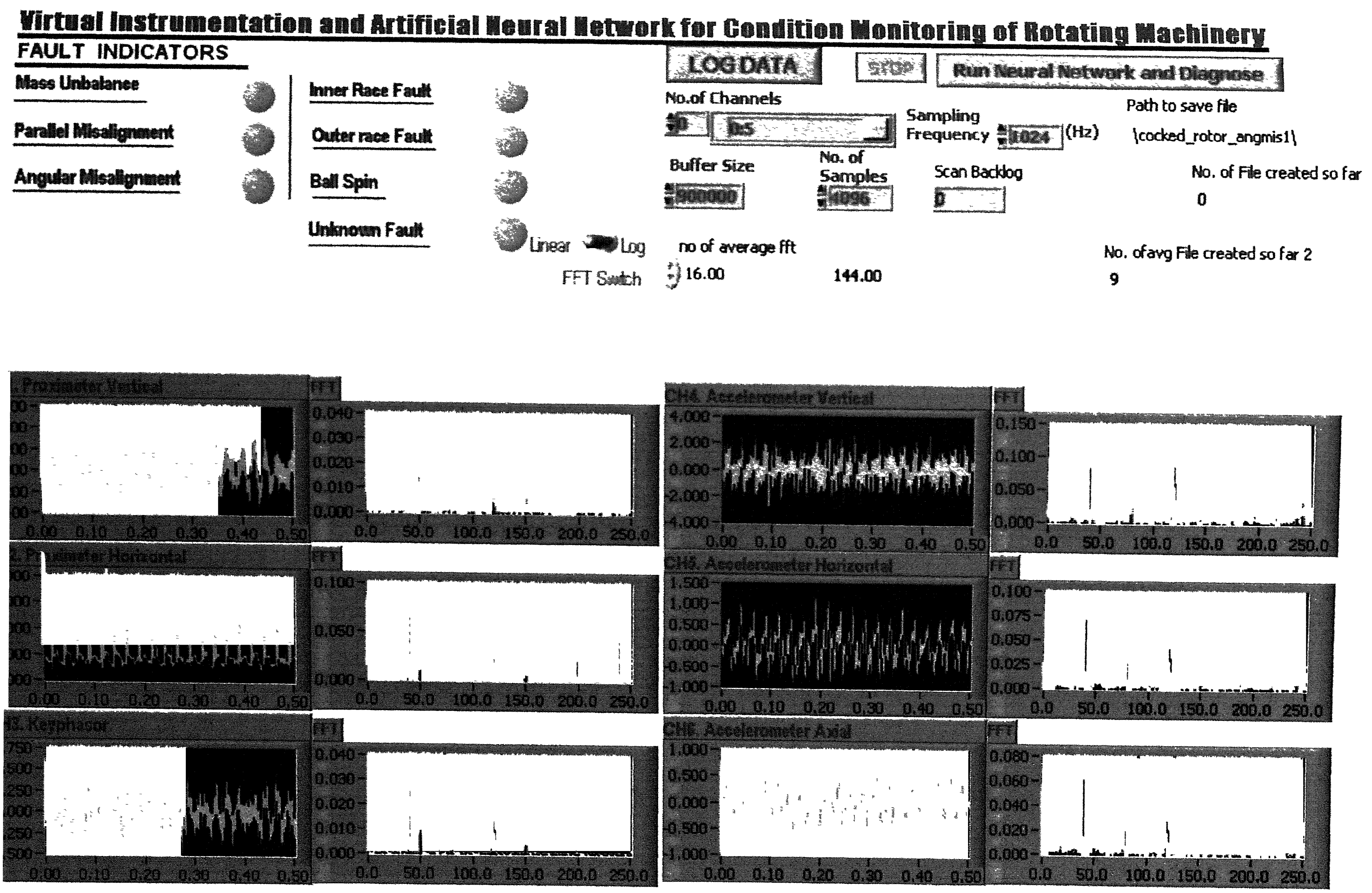
(xvi) Ball Spin + Unbalance

(xvii) Ball Spin +Gear Box attached

(xviii) Cocked Rotor +Parallel Misalignment

(xix) Cocked Rotor+ Unbalance

(xx) Reciprocating +Unbalance



Virtual Instrumentation and Artificial Neural Network for Condition Monitoring of Rotating Machinery

FAULT INDICATORS

Mass Unbalance

Parallel Misalignment

Angular Misalignment

Inner Race Fault

Outer race Fault

Ball Spin

Unknown Fault

FFT Switch: Linear / Log

LOG DATA

STOP

Run Neural Network and Diagnose

No.of Channels: 0, lb5

Sampling Frequency: 1024 (Hz)

Path to save file: \cocked_rotor_angmis1\

Buffer Size: 900000

No. of Samples: 4096

Scan Backlog: 0

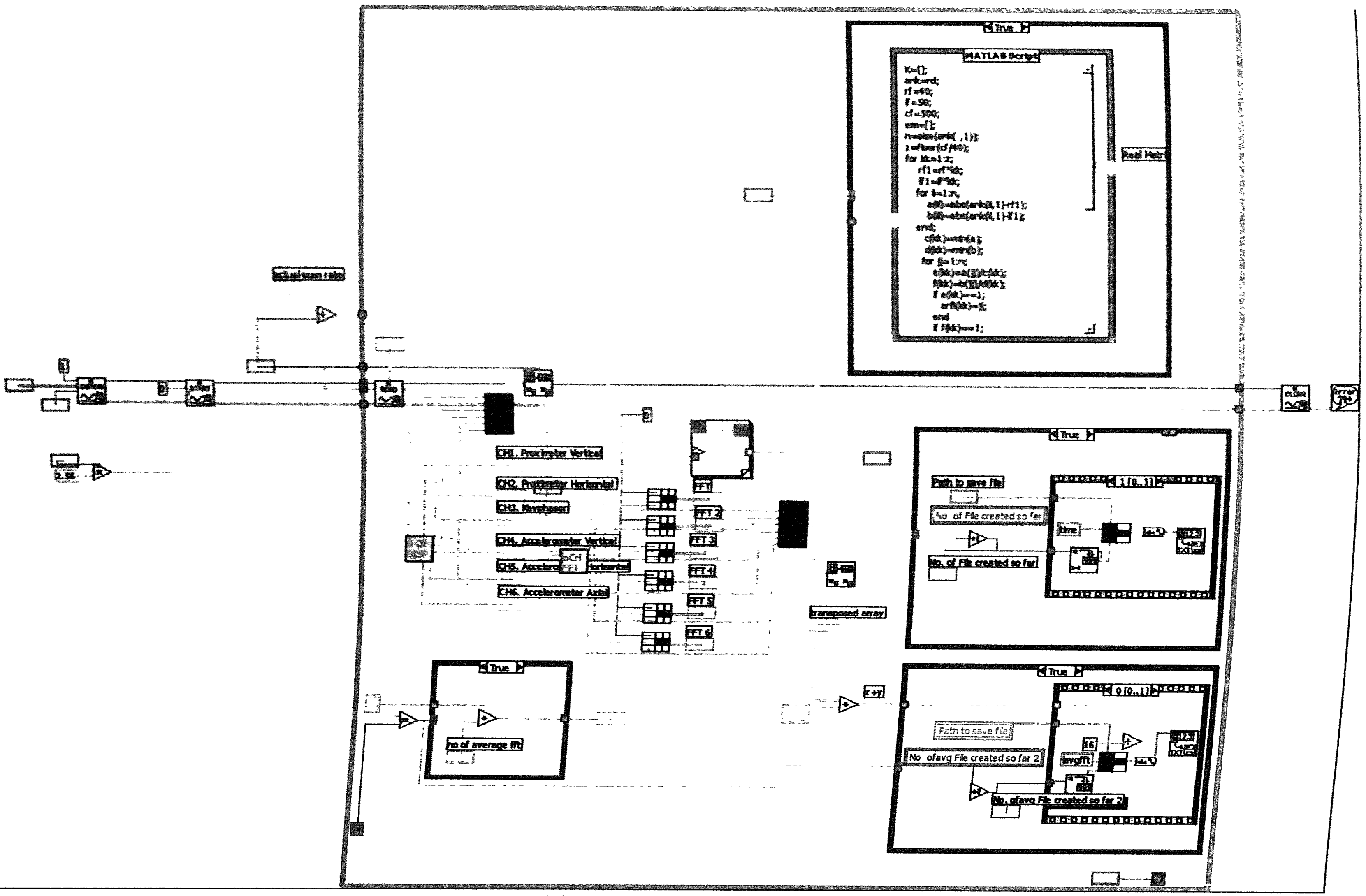
No. of File created so far: 0

no of average fft: 16.00

144.00

No. ofavg File created so far 2

9



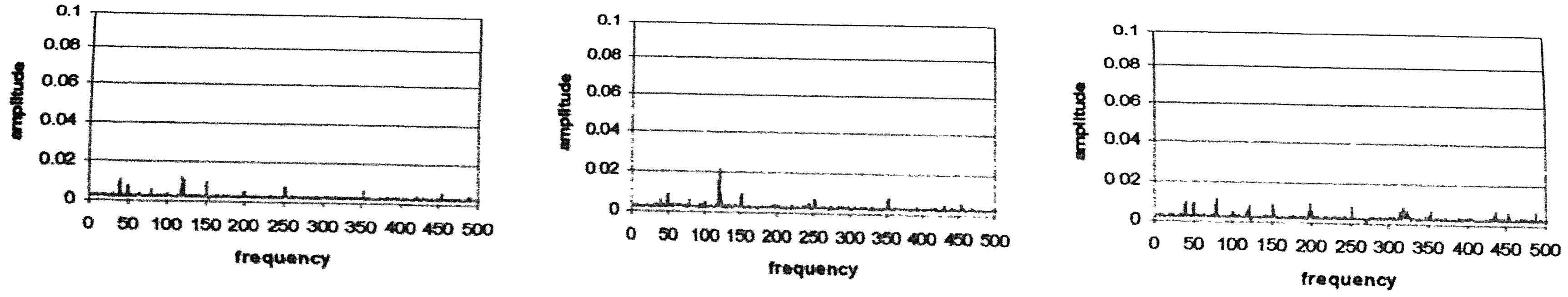
```
K=[];
ank=rd;
rf=40;
ff=50;
cf=500;
em=[];
n=size(ank( ,1));
z=floor(cf/40);
for kk=1:z;
    rf1=rf*kk;
    ff1=ff*kk;
   for ii=1:n,
      a(ii)=abs(ank(ii,1)-rf1);
      b(ii)=abs(ank(ii,1)-ff1);
   end;
     c(kk)=min(a);
     d(kk)=min(b);
    for jj=1:n;
       e(kk)=a(jj)/c(kk);
       f(kk)=b(jj)/d(kk);
       if e(kk)==1;
          arf(kk)=jj;
       end
       if f(kk)==1;
```

(b) Block diagram of the VI

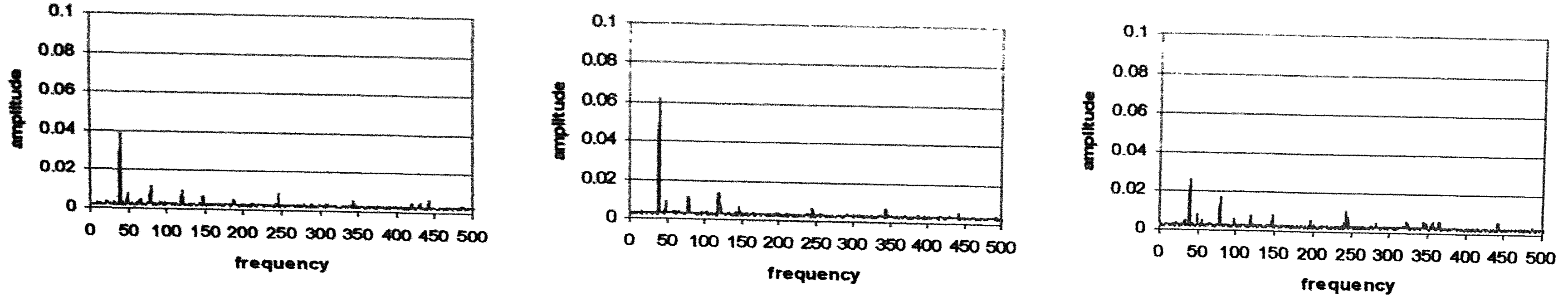
Forty five sets of vibration signals were acquired for each of the above seven separate faults and twenty mixed faults. Each set is average over 16 signals. Each set includes signals from the six sensors mentioned earlier. Out of the forty five sets, thirty five sets of each fault were employed to train the network and the remaining ten were set aside for validation of the trained network. Time domain vibration signals were acquired at a scan rate of 2621 samples/sec. Frequency domain transformation was carried out on line using an FFT algorithm available in LabVIEW.

Typical FFT data for some faults are shown in the Figs. 4.5 .The figures shows FFT of signals obtained from vertical horizontal and axial accelerometers .It can clearly be seen from the figures that the spectrum of each fault is unique with its own distinguishing peaks and shape. Values at various harmonics of the running frequency can be used for identifying the unknown fault.

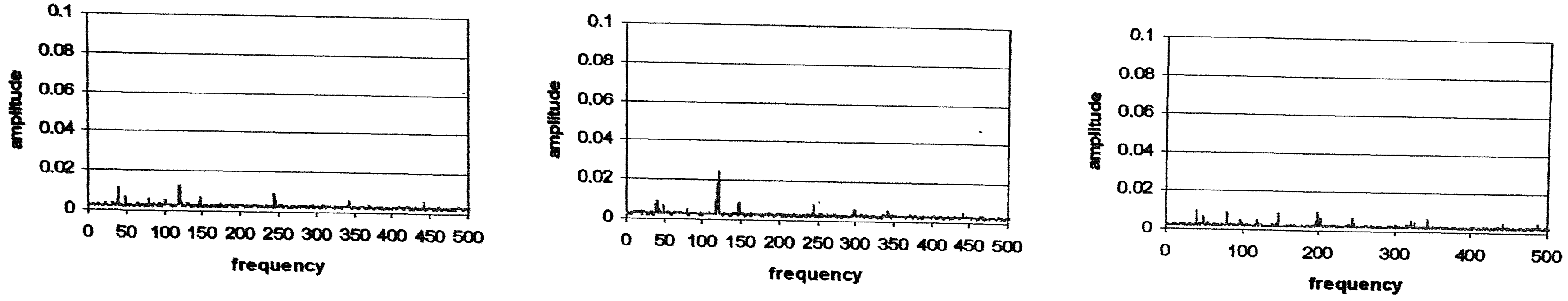
These FFT signals are further processed to prepare inputs for Neural Network training. This is described in Chapter 5.



(a) No Fault



(b) Mass Unbalalance

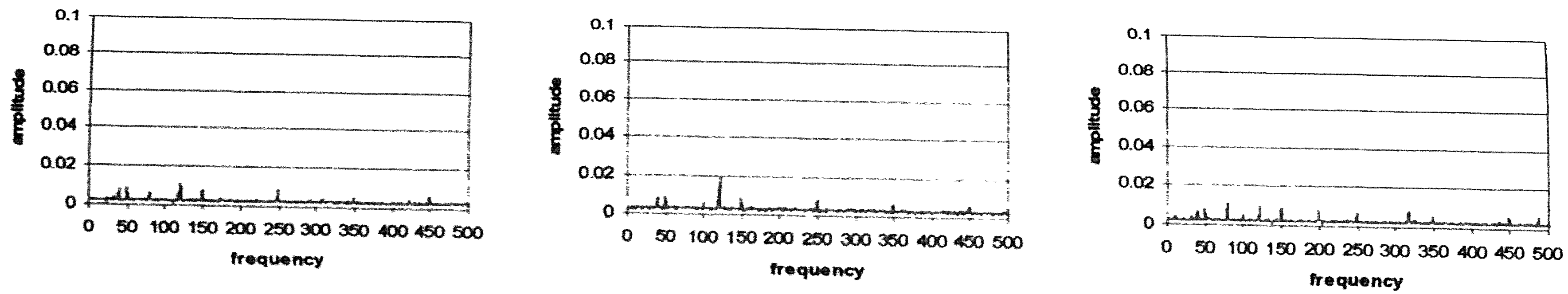


Vertical Accelerometer

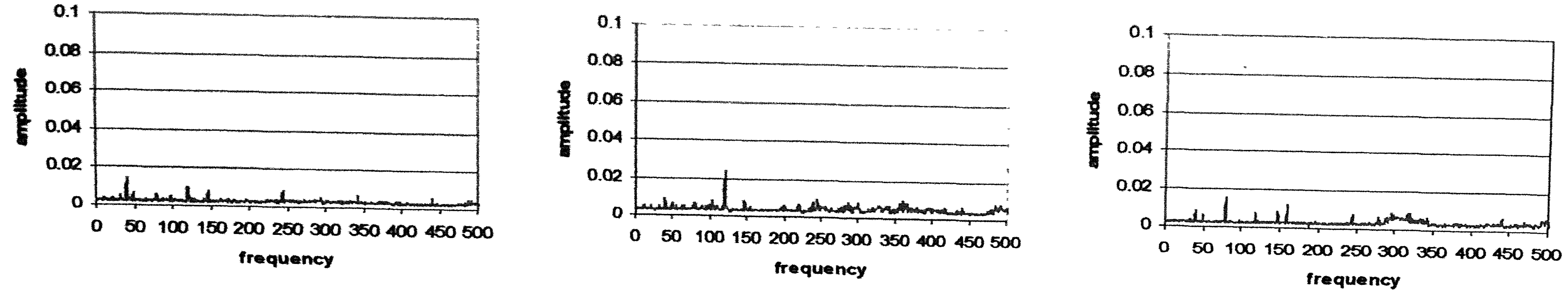
Horizontal Accelerometer

Axial Accelerometer

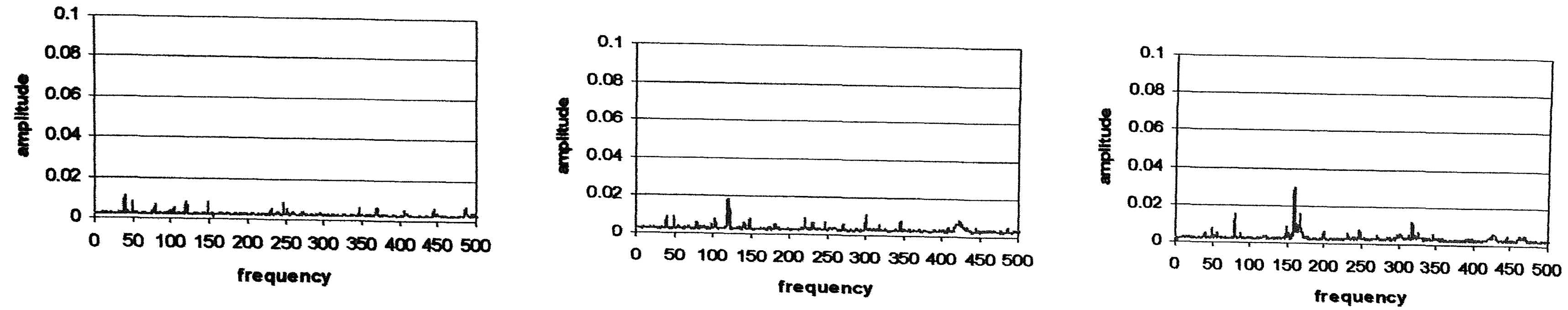
(c) Parallel Misalignment



(d) Angular Misalignment



(e) Bearing Inner Race Fault



Vertical Accelerometer Horizontal Accelerometer Axial Accelerometer

(f) Bearing Outer Race Fault

# CHAPTER 5

# NETWORK TRAINING AND VALIDATION

For the present work seven separate faults and twenty combinations of faults were introduced in the test rig. For the faults under consideration the time domain and the frequency domain data was acquired and the digital data is stored in separate files. For each fault forty five signals were recorded out of which thirty five sets were used for training of the networks and the balance ten sets were set aside for validation of the trained networks. Each signal stored in file is average of sixteen signals. Data in its raw form cannot be used for the training of a neural network as the amount of data involved is very large and it would be computationally demanding and cumbersome to carry out trials with various network architectures.

From the various signals parameters containing information, which is considered are to be extracted. These parameters are called the diagnosis features. Extraction of significant features from raw data is crucial to any condition monitoring exercise. Extraction of these features from a vibration signal can be sometimes arduous since they are almost always contaminated with noise. However in the present work, since the data is being collected under laboratory conditions from a commercially available test rig that is custom made for such an exercise, the noise level is very low. Therefore there was no requirement for noise removal from the signals. Table 5.1 shows the fault numbers assigned to various faults which we will use to identify the faults.

## 5.1 Feature Extraction

Feature extraction from vibration signatures to characterize particular machine faults has received considerable attention from researchers. Elkordy, Chang and Lee (1994) used state variables computed from the strain mode shapes and displacement mode shapes for structural damage diagnosis and condition monitoring of multi-storied buildings. Canonical correlation analysis was used by Mayes (1994) to study oil-whirl problem and reactive load dependent behavior of power generator. McCormick and Nandi (1997) extracted time-invariant features from stationary time series by estimating zero-lag moments. However, these procedures consume more computational time and memory since the moments and cross-moments need to be calculated.

FFT is widely recognized as a powerful tool to characterize most of the common faults in rotating machinery. Frequency spectrum is most commonly used and understood by most of the vibration professionals. Also, frequency domain data can be easily obtained from time series since a number of FFT analyzers, both digital and analogue, are available off the shelf. However, randomness in the frequency spectrum along with shifting frequencies does not allow simple diagnosis. The algorithm is developed for detection of frequencies which are significant for particular faults. The algorithm can take the frequency domain spectrum as input and provide the amplitudes of the frequency signals at various harmonics of the running speed.

The algorithm was written using MATLAB and functions in the following manner-

(1) The first column of frequency data stored in the file contains frequency. First of all we calculate the difference of all values of frequency to that running frequency and takes the absolute value of that. This absolute difference is stored in an array.

(2) We calculate the minimum number of that array and than index of that array.

(3) Once the index is obtained we stores the values of three previous and three latter values of that index in all channels in any variable.

(4) We calculate the maximum of above seven values for each channel.

(5) In the same way we calculate the maximum value near the higher harmonics.

The algorithm for preparing Input Vectors is shown in Fig. 5.1.

The program has the following user interactive controls:

(i) Rotational Speed of the machine.

(ii) The upper level of the harmonics up to which evaluation needs to be done.

(iii) The width of the frequency band on either side of the harmonics.

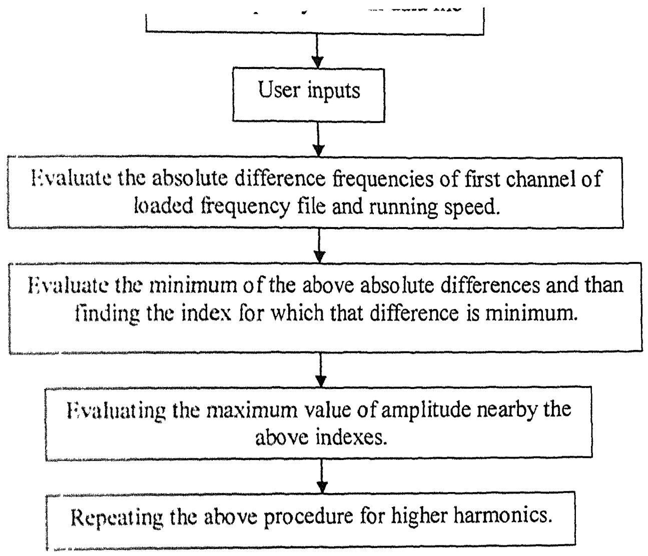

User inputs
Evaluate the absolute difference frequencies of first channel of loaded frequency file and running speed.
Evaluate the minimum of the above absolute differences and than finding the index for which that difference is minimum.
Evaluating the maximum value of amplitude nearby the above indexes.
Repeating the above procedure for higher harmonics.

**Fig. 5.1 Algorithm for preparation of input vector.**

The faults introduced in the rig are given numbers for easy reference later. The list of these fault numbers is given in Table 5.1.

**Table 5.1 Fault Numbering**

| Fault Name | Fault Number |
|---|---|
| No Fault | 1 |
| Mass Unbalance | 2 |
| Parallel Misalignment | 3 |
| Angular Misalignment | 4 |
| Bearing Inner Race Defect | 5 |
| Bearing Outer Race Defect | 6 |
| Bearing Ball Spin | 7 |
| Cocked Rotor | 8 |
| Loose Belt | 9 |
| Tight Belt | 10 |
| Reciprocating | 11 |
| Parallel Misalignment +Unbalance | 12 |
| Angular Misalignment +Unbalance | 13 |
| Inner Race +Unbalance | 14 |
| Outer Race +Unbalance | 15 |
| Ball Spin + Unbalance | 16 |
| Ball Spin +Gear Box Attached | 17 |
| Cocked Rotor +Parallel Misalignment | 18 |
| Cocked Rotor +Unbalance | 19 |
| Reciprocating +Unbalance | 20 |

FFT of the signals were recorded into files up to a frequency of 1048Hz. Automatic preparation of training vector was carried out using algorithm described earlier to compute 12 harmonics of the rotational speed and stored.

During this study, while measurements were taken with six sensors, which included both proximity pick-ups and accelerometers, it was found that shaft displacement signals acquired by proximity probes do not provide any additional information for the present rotor set up. Also, phase information from the key phasor was not utilized during this study. Therefore, further data processing was carried out only for the three accelerometer channels, namely (i) Bearing housing vertical acceleration (ii) Bearing housing horizontal acceleration (iii) Bearing housing axial acceleration. This reduction helps in reducing the computational effort considerably.

## 5.2 Creation of data sets for network training

For each run data from three channels vertical accelerometer, horizontal accelerometer and axial accelerometer are employed for training. For each channel we have a feature vector of 12 elements. This is shown in Table 5.2. Thus the feature vector for each data set is a vector of size 36.

**Table 5.2 Typical Harmonic Amplitudes from acceleration data (Not normalized)**

| | No Fault Condition | | | Mass Unbalance | | | Parallel Misalignment | | |
|---|---|---|---|---|---|---|---|---|---|
| | vertical | horizontal | axial | vertical | horizontal | axial | vertical | horizontal | axial |
| Amplitude Peaks at various Harmonics | 0.007 | 0.006 | 0.006 | 0.016 | 0.023 | 0.011 | 0.011 | 0.009 | 0.0080 |
| | 0.005 | 0.005 | 0.008 | 0.005 | 0.005 | 0.007 | 0.005 | 0.004 | 0.0070 |
| | 0.016 | 0.032 | 0.008 | 0.005 | 0.009 | 0.004 | 0.018 | 0.034 | 0.0060 |
| | 0.002 | 0.004 | 0.003 | 0.003 | 0.003 | 0.003 | 0.003 | 0.003 | 0.0030 |
| | 0.003 | 0.004 | 0.004 | 0.003 | 0.004 | 0.004 | 0.003 | 0.004 | 0.0040 |
| | 0.003 | 0.005 | 0.004 | 0.003 | 0.003 | 0.003 | 0.003 | 0.005 | 0.0040 |
| | 0.003 | 0.003 | 0.003 | 0.002 | 0.003 | 0.003 | 0.003 | 0.004 | 0.0040 |
| | 0.003 | 0.003 | 0.006 | 0.002 | 0.003 | 0.004 | 0.003 | 0.003 | 0.0050 |
| | 0.003 | 0.003 | 0.004 | 0.003 | 0.003 | 0.003 | 0.003 | 0.003 | 0.0030 |
| | 0.003 | 0.003 | 0.003 | 0.003 | 0.003 | 0.003 | 0.003 | 0.003 | 0.0030 |
| | 0.002 | 0.003 | 0.004 | 0.006 | 0.005 | 0.006 | 0.003 | 0.004 | 0.0050 |
| | 0.002 | 0.004 | 0.003 | 0.003 | 0.003 | 0.002 | 0.003 | 0.004 | 0.0030 |

Seven separate and twenty combined faults have been considered for the present work. For each fault 45 sets of signals were stored. Each set comprised of 3 acceleration signals, as described earlier, in the three directions (vertical, horizontal and axial). Out of these 45 sets for each fault, 35 sets were utilized for training and the remaining 10 were kept separately for testing of the trained networks.

Thus the training data set for seven separate faults comprises of two hundred and forty five vectors, each of size 36 and for twenty mixed faults comprises of seven hundred vectors each of size 36. MATLAB programs are written which can load all forty five files, extract features from these files, save vectors for each fault, and then prepare training data set size given above. Thus the training set size for seven faults is 36×245 and for twenty mixed faults it is 36×700 The data is normalized before sending it for network training. The data is normalized in the range [0.1, 0.9] using the following formula:

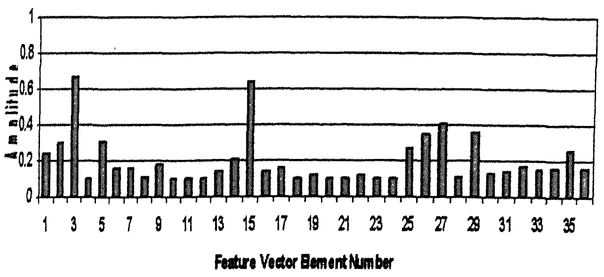
$$X_i = \left(\left(\frac{q_i - \min dat}{\max dat - \min dat}\right) * 0.8\right) + 0.1 \qquad (5.1)$$

where $X_i$ is the normalized value, $q_i$ is the un-normalized value, min dat is the minimum of all the elements going to the same neuron as the element being normalized and max dat is the maximum.

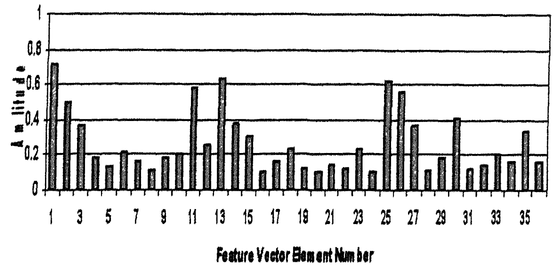
Figs 5.2 (a) to (t) show the plots of feature vectors for the various faults that are employed as inputs for the network training.

The target vector data set for back propagation is generated next. The target vector is chosen to be of the same size as the number of faults i.e. seven or twenty. The target vector for the first fault will have a value of 1 for its first element and all other elements will be zero. Similarly the target vector for the second vector will have its second element as 1 and rest of the elements as zero. Since the values of the target data set are only binary values, the data set can be easily prepared by a simple MATLAB code.

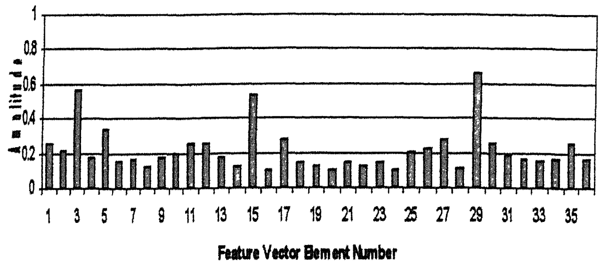
Typical training vectors created after extracting the features from the frequency domain data for seven faults is shown in Table 5.3. Back Propagation training is done with the



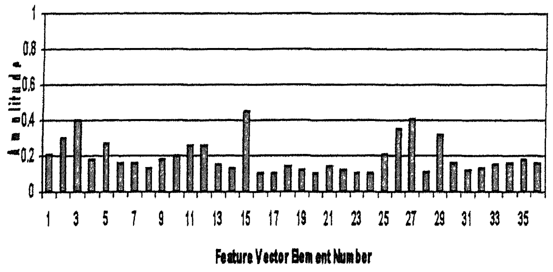
**(a) No Fault Condition**



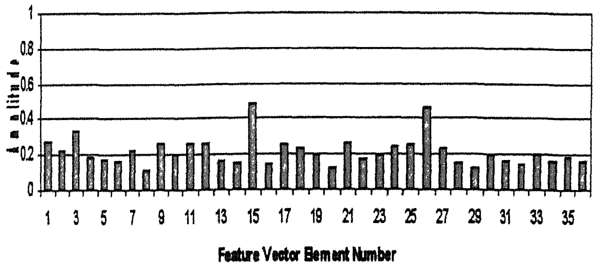
**(b) Mass Unbalance**



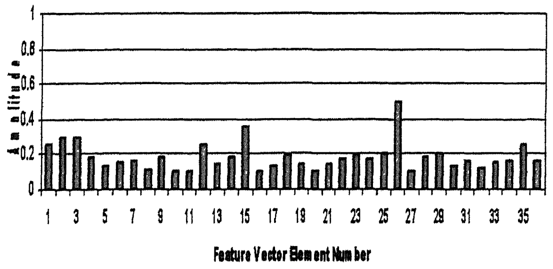
**(c) Parallel Misalignment**



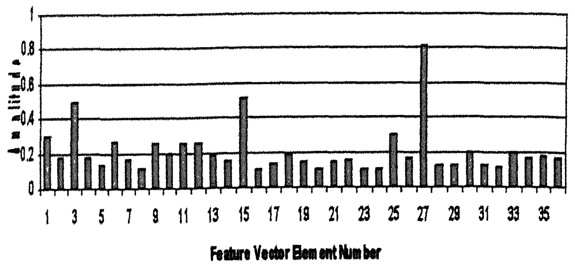
**(d) Angular Misalignment**



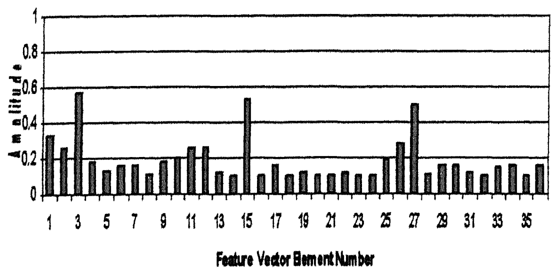
**(e) Inner Race Fault**



**(f) Outer Race Fault**

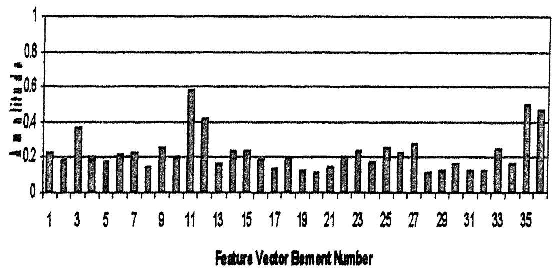


**(g) Bearing Ball Spin**

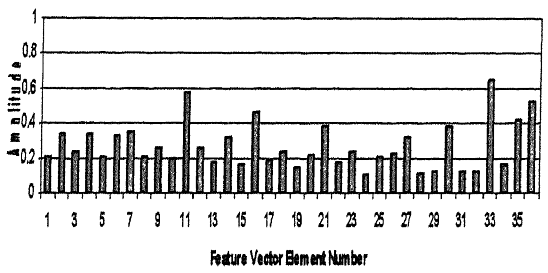


**(h) Cocked Rotor**

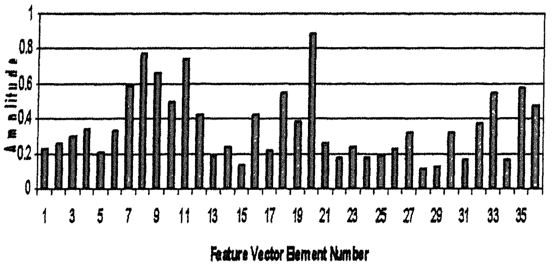
**Fig 5.2 Plots of normalized Feature Vectors for various fault conditions**



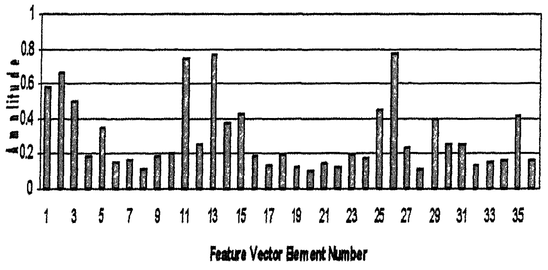
**(i) Loose Belt**



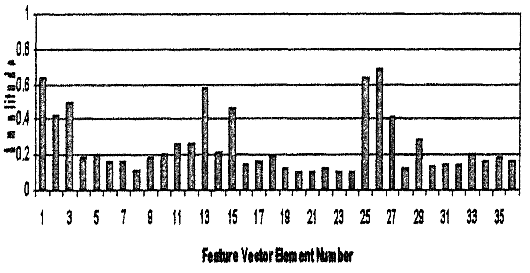
**(j) Tight Belt**



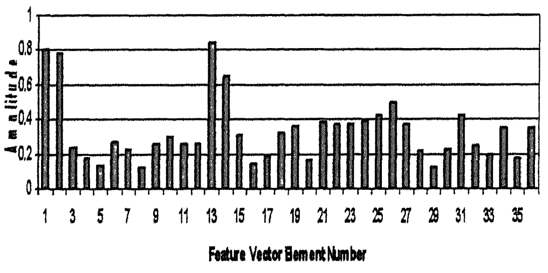
**(k) Reciprocating Mechanism Attached**



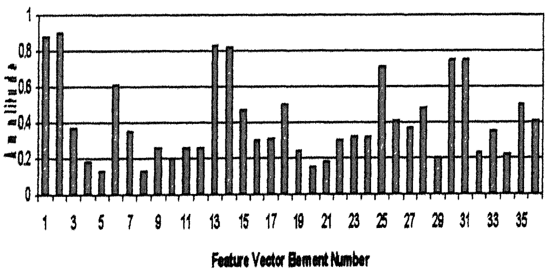
**(l) Parallel Misalignment +Unbalance**



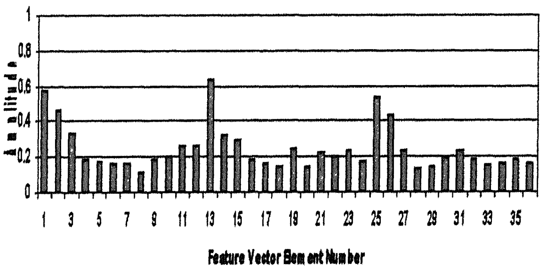
**(m) Angular Misalignment + Unbalance**



**(n) Inner Race Fault + Unbalance**

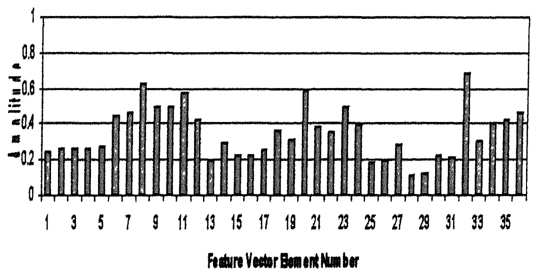


**(o) Outer Race Fault + Unbalance**

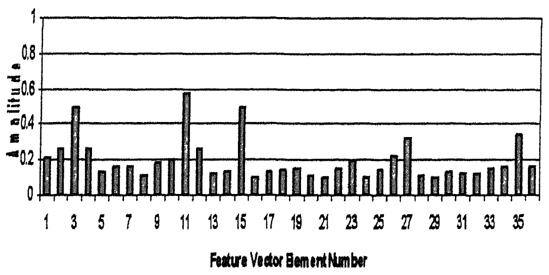


**(p) Ball-Spin + Unbalance**

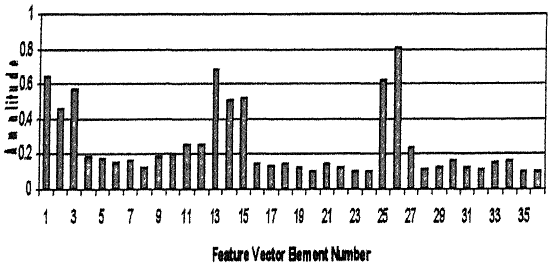
**Fig 5.2 (Contd.) Plots of normalized Feature Vectors for various fault conditions**



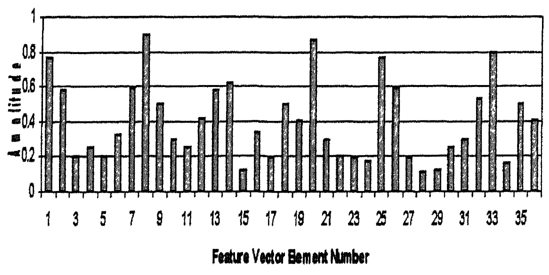
**(q) Ball-Spin+Gear box attached**



**(r) Cocked Rotor +Parallel Misalignment**



**(s) Cocked Rotor +Unbalance**



**(t) Unbalance + Reciprocating mechanism attached**

**Fig 5.2 (Contd.) Plots of normalized Feature Vectors for various fault conditions**

above data only while the Self Organizing training is done after normalizing the vectors between 0.1 to 0.9.Training of Back Propagation is done keeping the vectors of each faults one by one while the training of self organizing is done keeping all the vectors of a single fault at one place only. Table 5.4 shows the target vectors used for training for back propagation.

Table5.3 Typical Training Input Vector for seven faults used for back Propagation Network

| | Fault Numbers | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Amplitude Peaks of first 12 Harmonics of Vertical, Horizontal And Axial accelerometers (mV) | 7 | 16 | 11 | 6 | 12 | 8 | 7 | 11 | 41 | 15 | 10 | 14 | 12 | 15 | 11 | 43 | 13 | 8 | 14 | 11 | 15 |
| | 5 | 5 | 5 | 5 | 4 | 5 | 3 | 7 | 11 | 5 | 6 | 4 | 7 | 5 | 6 | 13 | 6 | 7 | 5 | 7 | 4 |
| | 16 | 5 | 18 | 8 | 7 | 6 | 6 | 26 | 16 | 15 | 12 | 8 | 8 | 7 | 22 | 14 | 16 | 12 | 9 | 8 | 10 |
| | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 14 | 3 | 3 | 9 | 3 | 3 | 3 | 25 | 3 | 4 | 17 | 4 | 3 | 3 |
| | 3 | 3 | 3 | 3 | 4 | 3 | 4 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 3 | 4 |
| | 3 | 2 | 3 | 4 | 4 | 4 | 3 | 4 | 3 | 3 | 3 | 4 | 3 | 3 | 4 | 3 | 2 | 3 | 5 | 3 | 3 |
| | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 5 | 2 | 4 | 5 | 4 | 3 | 3 | 3 | 3 | 3 | 4 | 4 | 3 | 3 |
| | 3 | 3 | 3 | 3 | 4 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 3 | 3 | 3 | 4 | 3 | 3 |
| | 3 | 3 | 3 | 3 | 4 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| | 2 | 6 | 3 | 3 | 3 | 3 | 3 | 3 | 5 | 3 | 3 | 3 | 3 | 3 | 3 | 6 | 3 | 3 | 3 | 3 | 3 |
| | 2 | 3 | 3 | 3 | 4 | 3 | 3 | 3 | 6 | 3 | 3 | 3 | 3 | 5 | 3 | 4 | 3 | 3 | 4 | 3 | 3 |
| | 6 | 23 | 9 | 6 | 7 | 6 | 7 | 7 | 63 | 9 | 9 | 8 | 8 | 13 | 8 | 63 | 9 | 8 | 8 | 8 | 12 |
| | 5 | 5 | 4 | 3 | 4 | 5 | 4 | 6 | 10 | 4 | 4 | 6 | 8 | 5 | 7 | 12 | 4 | 4 | 7 | 6 | 4 |
| | 32 | 9 | 34 | 19 | 19 | 13 | 9 | 51 | 25 | 30 | 31 | 23 | 18 | 14 | 46 | 20 | 30 | 30 | 25 | 19 | 26 |
| | 4 | 3 | 3 | 3 | 5 | 3 | 4 | 4 | 3 | 4 | 3 | 5 | 3 | 4 | 3 | 3 | 3 | 3 | 5 | 3 | 3 |
| | 4 | 4 | 4 | 6 | 9 | 5 | 4 | 8 | 5 | 5 | 12 | 7 | 5 | 7 | 9 | 4 | 5 | 11 | 8 | 4 | 5 |
| | 5 | 3 | 5 | 5 | 5 | 3 | 4 | 5 | 4 | 3 | 5 | 5 | 3 | 6 | 4 | 3 | 3 | 4 | 5 | 4 | 7 |
| | 3 | 3 | 4 | 3 | 8 | 4 | 4 | 3 | 3 | 3 | 3 | 8 | 3 | 5 | 3 | 3 | 3 | 3 | 6 | 4 | 4 |
| | 3 | 3 | 3 | 3 | 8 | 6 | 5 | 4 | 3 | 3 | 4 | 8 | 6 | 4 | 5 | 3 | 3 | 4 | 7 | 5 | 5 |
| | 3 | 3 | 3 | 3 | 8 | 4 | 5 | 3 | 3 | 4 | 4 | 6 | 4 | 4 | 3 | 3 | 4 | 4 | 6 | 4 | 4 |
| | 3 | 3 | 3 | 3 | 7 | 4 | 5 | 3 | 3 | 3 | 3 | 6 | 5 | 4 | 3 | 3 | 3 | 3 | 5 | 4 | 5 |
| | 3 | 5 | 4 | 4 | 5 | 4 | 3 | 4 | 6 | 4 | 4 | 4 | 4 | 4 | 3 | 5 | 4 | 4 | 4 | 5 | 4 |
| | 4 | 3 | 4 | 3 | 5 | 4 | 4 | 3 | 3 | 4 | 3 | 5 | 4 | 4 | 4 | 3 | 4 | 3 | 5 | 3 | 3 |
| | 6 | 11 | 8 | 5 | 7 | 4 | 5 | 9 | 27 | 8 | 7 | 8 | 7 | 11 | 10 | 27 | 8 | 8 | 8 | 7 | 11 |
| | 8 | 7 | 7 | 7 | 12 | 10 | 3 | 11 | 16 | 6 | 9 | 16 | 15 | 4 | 12 | 18 | 8 | 11 | 15 | 16 | 3 |
| | 8 | 4 | 6 | 8 | 7 | 3 | 4 | 13 | 9 | 6 | 11 | 7 | 3 | 14 | 15 | 10 | 7 | 11 | 7 | 3 | 13 |
| | 3 | 3 | 3 | 3 | 7 | 13 | 3 | 3 | 3 | 3 | 3 | 8 | 14 | 3 | 3 | 3 | 3 | 3 | 8 | 9 | 4 |
| | 4 | 4 | 4 | 4 | 3 | 5 | 3 | 21 | 8 | 5 | 16 | 3 | 7 | 4 | 42 | 7 | 11 | 32 | 4 | 7 | 3 |
| | 4 | 3 | 4 | 4 | 4 | 4 | 3 | 6 | 3 | 4 | 5 | 4 | 4 | 6 | 6 | 8 | 5 | 7 | 5 | 4 | 5 |
| | 3 | 3 | 4 | 3 | 6 | 7 | 3 | 7 | 4 | 4 | 6 | 7 | 6 | 4 | 10 | 3 | 3 | 7 | 5 | 9 | 3 |
| | 6 | 4 | 5 | 8 | 7 | 10 | 3 | 13 | 6 | 6 | 13 | 9 | 8 | 3 | 15 | 7 | 7 | 10 | 9 | 7 | 4 |
| | 4 | 3 | 3 | 3 | 3 | 4 | 3 | 4 | 4 | 3 | 3 | 3 | 5 | 3 | 6 | 3 | 3 | 5 | 4 | 4 | 3 |
| | 3 | 3 | 3 | 3 | 3 | 4 | 3 | 5 | 3 | 3 | 4 | 3 | 3 | 3 | 7 | 3 | 3 | 5 | 3 | 3 | 3 |
| | 4 | 6 | 5 | 4 | 3 | 4 | 3 | 7 | 8 | 4 | 5 | 3 | 5 | 3 | 4 | 6 | 4 | 4 | 3 | 5 | 3 |
| | 3 | 2 | 3 | 3 | 4 | 5 | 3 | 3 | 3 | 3 | 2 | 3 | 5 | 3 | 6 | 2 | 3 | 4 | 4 | 5 | 3 |

Table5.4 Typical Target Vector for Seven faults Used for Back Propagation Networks

| | Fault Numbers | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Output Vectors | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

## 5.3 Back Propagation Training and Results:-

The training vector for back propagation is prepared as described earlier keeping the vector of each fault one by one. In the same way the target vector also. Number of rows in the target vector is same as the number of faults. The training is carried first for seven separate faults and than twenty faults. The following architectures were tried out:

(i) 96-84-39-16

(ii) 96-72-48-24-16

(iii) 96-76-56-36-16

(iv) 96-84-56-28-16

(v) 96-80-56-32-16

(vi) 96-84-66-48-30-16

Training was carried out for each of the above architectures using Logistic Sigmoid activation function (LOGSIG) for separate seven faults. The back propagation training scheme used was TRAINRP. The convergence patterns of these networks are shown in Figs. 5.3 (a-f). The figures show that while training seven faults than the performance goal is obtained very early between 100 to 150 epochs for all of the architecture.

The training for twenty faults is done using Logistic Sigmoid activation function in the hidden layers and pure linear activation function in the output layer. The back propagation training scheme used was TRAINRP. Back Propagation Training for twenty faults is tried for following architecture:-

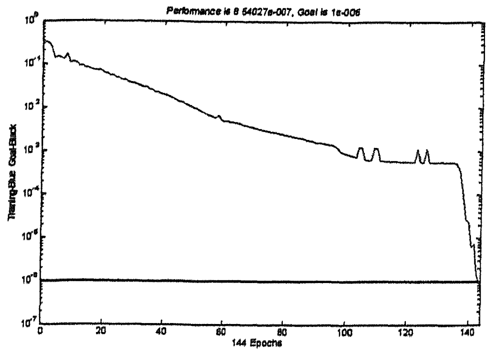
(i) 96-84-39-20

(ii) 96-72-48-24-20

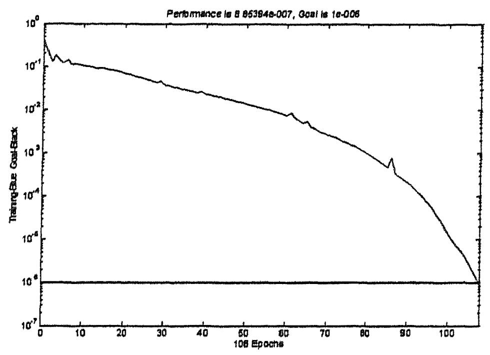
(iii) 96-84-56-28-20

(iv) 96-84-64-46-20

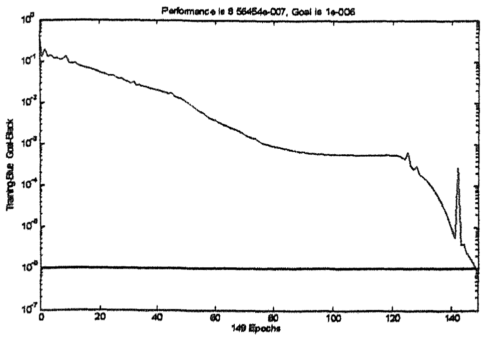
The training scheme for twenty faults is shown in Figs. 5.3 (g-j).From the training scheme it is clear that it takes large time to get desired goal even the goal is much higher than which is used in training the network for seven faults. Table 5.5 and table 5.6 shows the comparison for various architecture in terms of minimum error obtained and numbers of epochs to obtain the desired goal for seven faults and twenty faults respectively. Fig. 5.4 shows graphical representation of the minimum error obtained for various architecture for seven faults and Fig. 5.5 shows the graphical representation of total numbers of epochs for various architecture for seven faults. Fig.5.6 shows the minimum error obtained for various architecture for twenty faults and Fig 5.7 shows the number of epochs to obtain the desired goal for twenty faults. Table 5.7 shows the typical output vectors for seven faults when network is validated by validation data set of 10 vectors of each fault. Table 5.8 shows the typical output vectors for twenty faults when network is validated for by validation data set of 10 vectors for each fault.



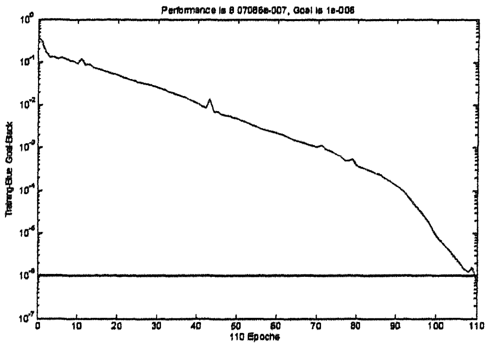
**(a) Network Architecture 96-84-39-7**



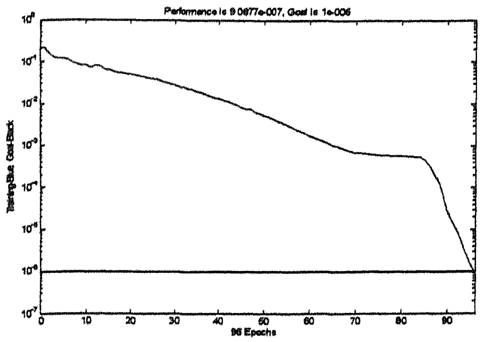
**b)Network Architecture 96-72-48-24-7**

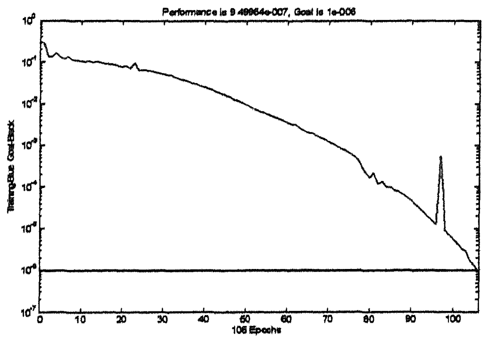


**(c)Network Architecture 96-76-56-36-7**



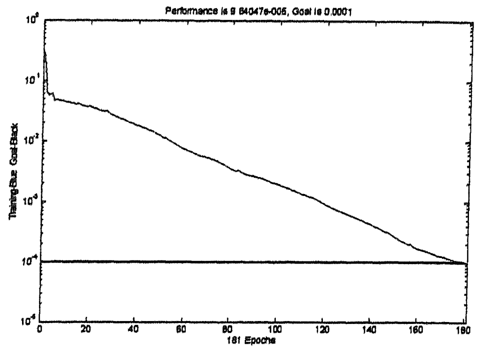
**(d) Network Architecture 96-84-56-28-7**



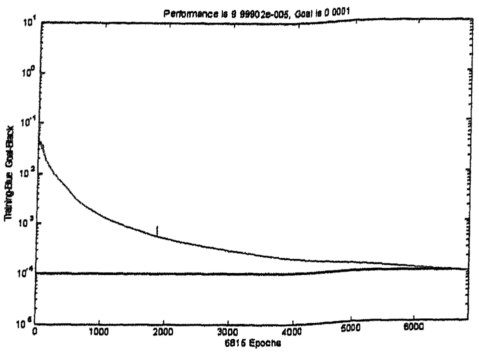


**(e) Network Architecture 96-80-56-32-7 (f) Network Architecture 96-84-66-48-30-7**

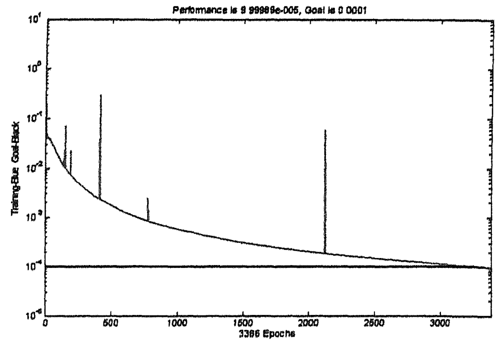
**Fig. 5.3 Convergence Patterns of Various Back Propagation Architecture**



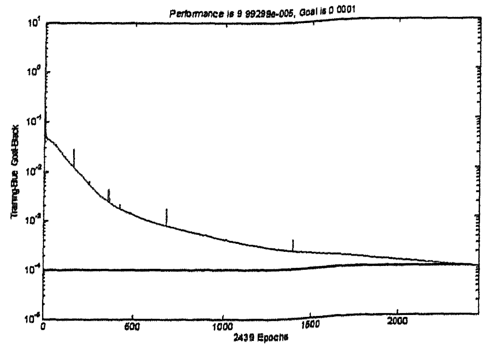
(g) Architecture 96-84-39-20



(h)Architecture 96-72-48-24-20



(i) Architecture 96-84-56-28-20



(j)Architecture 96-84-56-28-20

**Fig. 5.3(Contd.)** **Convergence Patterns of Various Back Propagation Architecture**

**Table5.5 Comparison of Various Back Propagation Networks for seven faults**

| SL.No | Network Architecture | Min. Error Achieved | No. of Epochs |
|---|---|---|---|
| 1 | 96-84-39-7 | 8.54 E -7 | 144 |
| 2 | 96-72-48-24-7 | 8.85 E -7 | 108 |
| 3 | 96-76-56-36-7 | 8.56 E-7 | 149 |
| 4 | 96-84-56-28-7 | 8.07 E-7 | 110 |
| 5 | 96-80-56-32-7 | 9.05 E-7 | 108 |
| 6 | 96-84-66-48-30-7 | 9.50 E -7 | 106 |

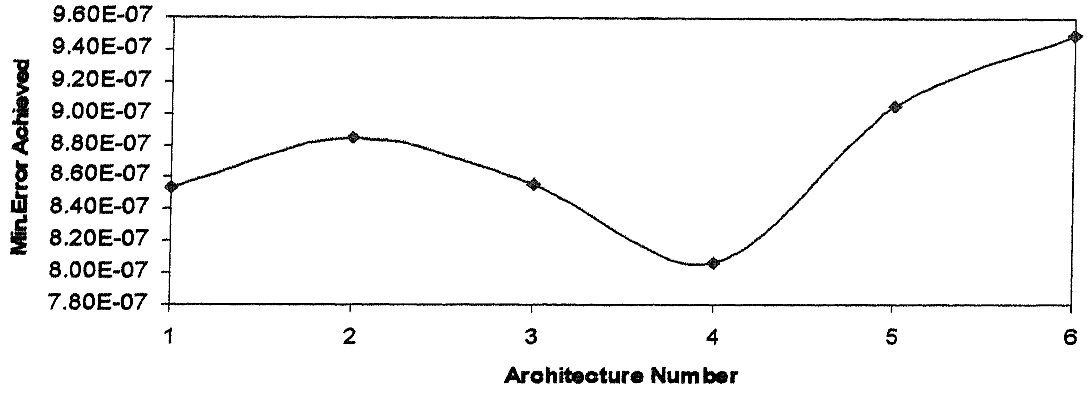


**Fig 5.4** Plot of minimum error achieved by various Network Architecture

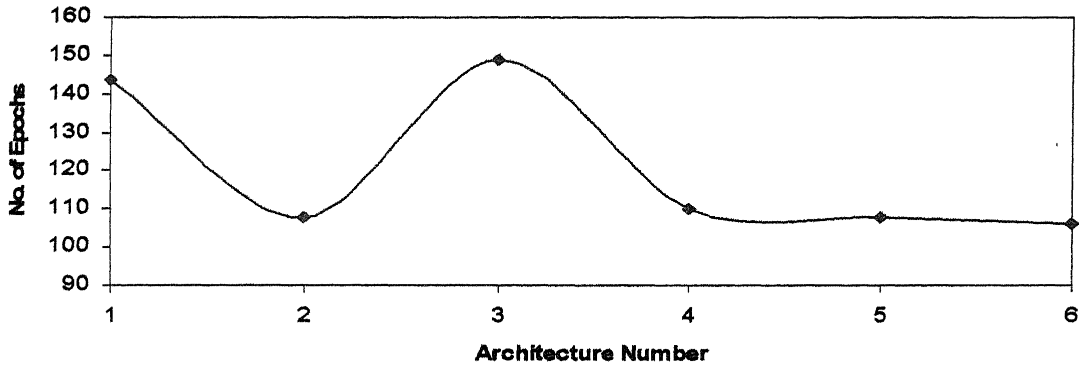


**Fig 5.5** Plot of epochs achieved by various Network Architecture

**Table5.6 Comparison of Various Back Propagation Networks for twenty faults**

| SL.No | Network Architecture | Min. Error Achieved | No. of Epochs |
|---|---|---|---|
| 1 | 96-84-39-20 | 9.84 E-5 | 181 |
| 2 | 96-72-48-24-20 | 9.99E-5 | 6815 |
| 3 | 96-84-56-28-20 | 9.99E-5 | 3386 |
| 4 | 96-84-64-46-20 | 9.99E-5 | 2439 |

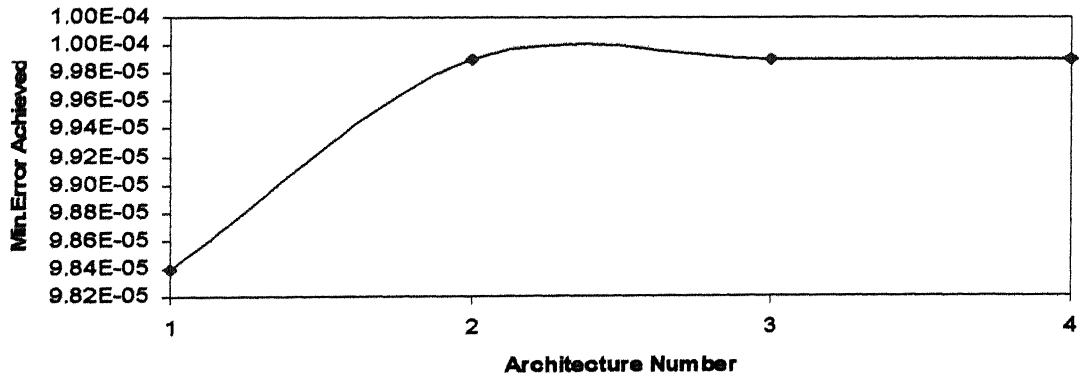


**Fig 5.6** Plot of minimum error achieved by various Network Architecture

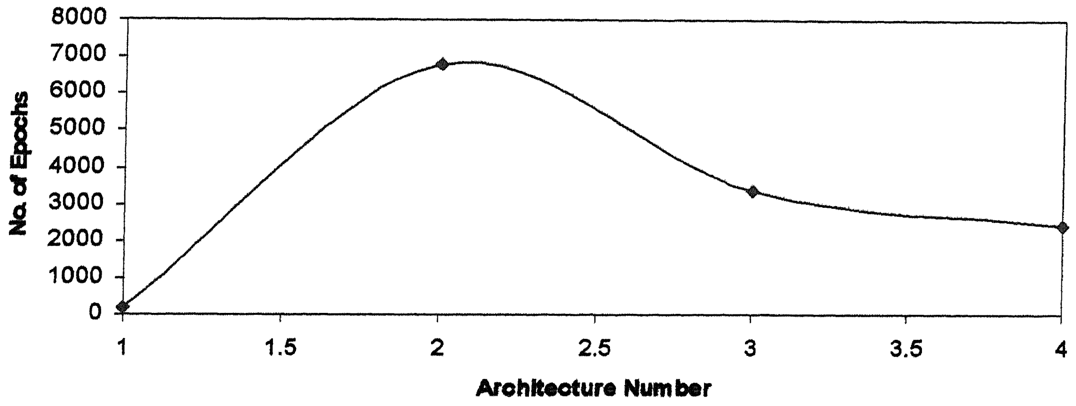


Fig 5.7 Plot of epochs achieved by various Network Architecture

Table 5.7 Typical Output Vector for seven faults for architecture 96-72-48-24-7 for training data set.

| | Fault Numbers | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Output Vectors | **0.997** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | **0.999** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.00 |
| | 0.000 | **1.000** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | **1.000** | 0.000 | 0.000 | 0.000 | 0.000 | 0.00 |
| | 0.000 | 0.001 | **1.000** | 0.000 | 0.000 | 0.000 | 0.000 | 0.001 | 0.000 | **1.000** | 0.000 | 0.000 | 0.000 | 0.00 |
| | 0.001 | 0.003 | 0.000 | **0.999** | 0.000 | 0.002 | 0.003 | 0.000 | 0.000 | 0.000 | **0.996** | 0.000 | 0.000 | 0.00 |
| | 0.000 | 0.000 | 0.000 | 0.000 | **1.000** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | **1.000** | 0.000 | 0.00 |
| | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | **1.000** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | **1.000** | 0.00 |
| | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | **0.998** | 0.003 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | **1.00** |

Table 5.8 Typical Output Vector for seven faults for architecture 96-72-48-24-7 for validation data set

| | Fault Numbers | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Output Vectors | **1.000** | 0.000 | 0.001 | 0.000 | 0.000 | 0.000 | 0.000 | **0.970** | 0.000 | 0.185 | 0.000 | 0.000 | 0.000 | 0.00 |
| | 0.000 | **1.000** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | **1.000** | 0.000 | 0.000 | 0.000 | 0.000 | 0.00 |
| | 0.002 | 0.000 | **1.000** | 0.000 | 0.000 | 0.000 | 0.000 | 0.288 | 0.000 | **0.839** | 0.000 | 0.000 | 0.000 | 0.00 |
| | 0.000 | 0.000 | 0.000 | **0.998** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | **0.983** | 0.000 | 0.000 | 0.00 |
| | 0.000 | 0.000 | 0.000 | 0.000 | **1.000** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | **1.000** | 0.000 | 0.00 |
| | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | **1.000** | 0.000 | 0.000 | 0.000 | 0.653 | 0.000 | 0.000 | **0.975** | 0.00 |
| | 0.000 | 0.000 | 0.000 | 0.000 | 0.001 | 0.000 | **0.997** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | **0.99** |

Table 5.9 Input for the Probabilistic Neural Network (20 faults)

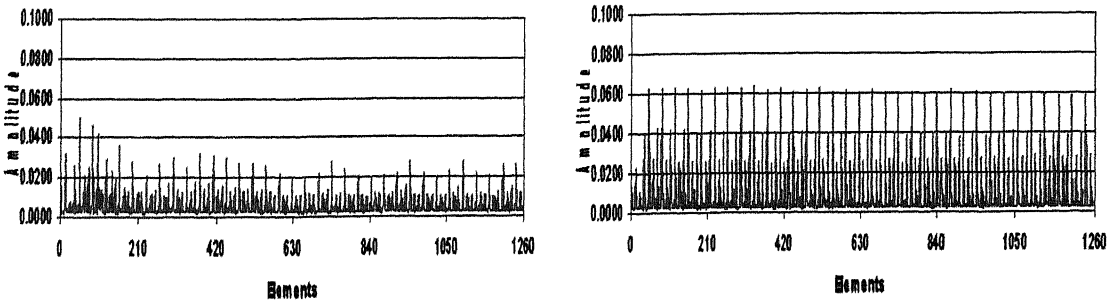
| | | | Fault Numbers | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** |
| Vector1(mV) | E-1 | **1** | 7 | 16 | 11 | 6 | 12 | 8 | 7 | 7 | 11 | 6 | 9 | 13 | 17 | 40 | 26 | 16 | 12 | 6 | 36 | 2 |
| | E-2 | **2** | 5 | 5 | 5 | 5 | 4 | 5 | 3 | 6 | 5 | 5 | 8 | 7 | 5 | 7 | 10 | 5 | 7 | 7 | 10 | 2 |
| | | . | | | | | | | | | | | | | | | | | | | | |
| | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | |
| | | . | . | . | . | . | . | . | . | . | . | . | | . | . | . | . | . | . | | | . |
| | E-36 | **36** | 3 | 2 | 3 | 3 | 4 | 5 | 3 | 2 | 7 | 7 | 8 | 3 | 3 | 5 | 5 | 4 | 7 | 3 | 3 | 2 |
| Vector 2(mV) | E-1 | **1** | 11 | 41 | 15 | 10 | 14 | 12 | 15 | 6 | 10 | 9 | 10 | 33 | 37 | 48 | 54 | 35 | 12 | 9 | 36 | 36 |
| | E-2 | **2** | 7 | 11 | 5 | 6 | 4 | 7 | 5 | 5 | 4 | 9 | 6 | 16 | 10 | 8 | 21 | 10 | 8 | 5 | 12 | 9 |
| | | . | | | | | | | | | | | | | | | | | | | | |
| | | . | . | . | . | . | . | . | . | . | | . | . | . | . | . | . | | . | . | . | . |
| | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| | E-36 | **36** | 3 | 3 | 3 | 2 | 3 | 5 | 3 | 3 | 8 | 10 | 8 | 3 | 4 | 5 | 8 | 5 | 7 | 3 | 2 | 4 |
| | | **1** | 11 | 43 | 13 | 8 | 14 | 11 | 15 | 13 | 11 | 11 | 10 | 32 | 35 | 46 | 54 | 34 | 11 | 6 | 36 | 45 |
| | | **2** | 6 | 13 | 6 | 7 | 5 | 7 | 4 | 5 | 4 | 8 | 5 | 15 | 10 | 15 | 21 | 8 | 7 | 6 | 12 | 14 |
| | | . | | | | | | | | | | | | | | | | | | | | |
| | | . | | | | | | | | | | | | | | | | | | | | |
| | | . | | | | | | | | | | | | | | | | | | | | |
| | | **36** | 12 | 42 | 15 | 9 | 13 | 12 | 16 | 16 | 11 | 9 | 11 | 32 | 38 | 49 | 54 | 34 | 9 | 11 | 37 | 46 |
| | | **1** | 6 | 11 | 6 | 7 | 5 | 7 | 4 | 6 | 5 | 6 | 6 | 15 | 10 | 13 | 21 | 10 | 7 | 6 | 12 | 16 |
| | | **2** | 14 | 11 | 15 | 12 | 8 | 9 | 14 | 17 | 8 | 9 | 6 | 14 | 14 | 6 | 10 | 9 | 6 | 13 | 16 | 6 |
| | | . | | | | | | | | | | | | | | | | | | | | |
| | | . | | | | | | | | | | | | | | | | | | | | |
| | | . | | | | | | | | | | | | | | | | | | | | |
| | | . | | | | | | | | | | | | | | | | | | | | |
| | | **36** | 3 | 3 | 2 | 3 | 4 | 3 | 3 | 2 | 9 | 12 | 11 | 3 | 3 | 10 | 11 | 5 | 8 | 2 | 3 | 9 |
| Vector 35(mV) | | **1** | 10 | 40 | 12 | 9 | 12 | 11 | 17 | 13 | 12 | 11 | 12 | 33 | 34 | 45 | 51 | 34 | 13 | 23 | 38 | 34 |
| | | **2** | 7 | 11 | 6 | 8 | 5 | 7 | 5 | 7 | 5 | 8 | 6 | 16 | 10 | 19 | 22 | 12 | 8 | 6 | 10 | 17 |
| | | . | | | | | | | | | | | | | | | | | | | | |
| | | . | | | | | | | | | | | | | | | | | | | | |
| | | . | | | | | | | | | | | | | | | | | | | | |
| | E-36 | **36** | 3 | 3 | 3 | 3 | 5 | 3 | 3 | 2 | 9 | 12 | 11 | 3 | 3 | 11 | 12 | 5 | 10 | 2 | 3 | 8 |

| Vector No. | No Fault |  |  | Mass Unbalance |  |  | Misalignment |  |  | Misalignment |  |  | Inner Race Fault |  |  | Outer Race fault |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  | 1 | 2 | . | 1 | 2 | . | 1 | 2 | . |  |  |  |  |  |  | 1 | 2 | . |
| Amplitude | 0.238 | . | . | 0.715 | . | . | 0.254 | . | . | 0 208 | . | . | 0.269 | . | . | 0.254 | . | . |
|  | 0.300 | . | . | 0.500 | . | . | 0.220 | . | . | 0.300 | . | . | 0.220 | . | . | 0.300 | . | . |
|  | 0.667 | . | . | 0.367 | . | . | 0.567 | . | . | 0.400 | . | . | 0.333 | . | . | 0.300 | . | . |
|  | 0.100 | . | . | 0.180 | . | . | 0.180 |  |  | 0.180 | . | . | 0.180 | . | . | 0.180 | . | . |
|  | 0.309 | . | . | 0.135 | . | . | 0.343 | . | . | 0.274 | . | . | 0.170 | . | . | 0.135 | . | . |
|  | 0.157 | . | . | 0.214 | . | . | 0.157 | . | . | 0.157 | . | . | 0.157 | . | . | 0.157 | . | . |
|  | 0.162 | . | . | 0.162 | . | . | 0.162 | . | . | 0.162 | . | . | 0.223 | . | . | 0.162 | . | . |
|  | 0.113 | . | . | 0.113 | . | . | 0.126 | . | . | 0.126 | . | . | 0.113 | . | . | 0.113 | . | . |
|  | 0.180 | . | . | 0.180 | . | . | 0.180 | . | . | 0.180 | . | . | 0.260 | . | . | 0.180 | . | . |
|  | 0.100 | . | . | 0.200 | . | . | 0.200 | . | . | 0.200 | . | . | 0.200 | . | . | 0.100 |  | . |
|  | 0.100 | . | . | 0.580 | . | . | 0.260 | . | . | 0.260 | . | . | 0.260 | . | . | 0 100 | . | . |
|  | 0.100 | . | . | 0.260 | . | . | 0.260 | . | . | 0.260 | . | . | 0.260 | . | . | 0.260 | . |  |
|  | 0.136 | . | . | 0.633 | . | . | 0.171 | . | . | 0.153 | . | . | 0.162 | . | . | 0.144 | . | . |
|  | 0.210 | . | . | 0.376 | . | . | 0.128 | . | . | 0.128 | . | . | 0.155 | . | . | 0.183 | . |  |
|  | 0.639 | . | . | 0.309 | . | . | 0.535 | . | . | 0.448 | . | . | 0.483 | . | . | 0.361 | . | . |
|  | 0.140 | . | . | 0.100 | . | . | 0.100 | . | . | 0.100 | . | . | 0.140 | . | . | 0.100 | . | . |
|  | 0.159 | . | . | 0.159 | . | . | 0.278 | . | . | 0 100 | . | . | 0.248 | . | . | 0.130 | . | . |
|  | 0.100 | . | . | 0.233 | . | . | 0.144 | . | . | 0.144 | . | . | 0.233 | . | . | 0.189 | . | . |
|  | 0.124 | . | . | 0.124 | . | . | 0.124 | . | . | 0.124 | . | . | 0.194 | . | . | 0.147 | . | . |
|  | 0.100 | . | . | 0.100 | . | . | 0.100 | . | . | 0.100 |  | . | 0.120 | . | . | 0.100 | . | . |
|  | 0.100 | . | . | 0.140 | . | . | 0.140 | . | . | 0.140 | . | . | 0.260 | . | . | 0.140 | . | . |
|  | 0.125 | . | . | 0.125 | . | . | 0.125 | . | . | 0.125 | . | . | 0.175 | . | . | 0.175 | . | . |
|  | 0.100 | . | . | 0.233 | . | . | 0.144 | . | . | 0.100 | . | . | 0.189 | . | . | 0.189 | . | . |
|  | 0.100 | . | . | 0.100 | . | . | 0.100 | . | . | 0.100 | . | . | 0.245 | . | . | 0.173 | . | . |
|  | 0.273 | . | . | 0.619 | . | . | 0.208 | . | . | 0.208 | . | . | 0.251 | . | . | 0.208 | . | . |
|  | 0.346 | . | . | 0.562 | . | . | 0.223 | . | . | 0.346 | . | . | 0.469 | . | . | 0.500 | . | . |
|  | 0.411 | . | . | 0.367 | . | . | 0.278 | . | . | 0.411 | . | . | 0.233 | . | . | 0.100 | . | . |
|  | 0.109 | . | . | 0.109 | . | . | 0.109 | . | . | 0.109 | . | . | 0.154 | . | . | 0.181 | . | . |
|  | 0.360 | . | . | 0.180 | . | . | 0.660 | . | . | 0.320 | . | . | 0.120 | . | . | 0.200 | . | . |
|  | 0.131 | . | . | 0.408 | . | . | 0.254 | . | . | 0.162 | . | . | 0.192 |  | . | 0.131 | . | . |
|  | 0.143 | . | . | 0.122 | . | . | 0.186 | . | . | 0.122 | . | . | 0.165 | . | . | 0.165 | . |  |
|  | 0.165 | . | . | 0.139 | . | . | 0.165 | . | . | 0.126 | . | . | 0.146 | . | . | 0.126 | . | . |
|  | 0.150 | . | . | 0.200 | . | . | 0.150 | . | . | 0.150 | . | . | 0.200 | . | . | 0.150 | . | . |
|  | 0.162 | . | . | 0.162 | . | . | 0.162 | . | . | 0.162 | . | . | 0.162 | . | . | 0.162 | . | . |
|  | 0.260 | . | . | 0.340 | . | . | 0.260 | . | . | 0.180 | . | . | 0.180 | . | . | 0.260 | . |  |

## 5.4 Probabilistic Neural Network Training

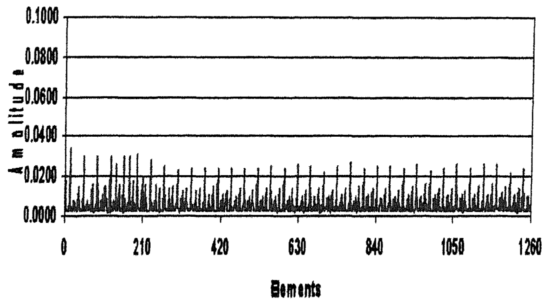
The probabilistic neural net work is trained for seven and twenty faults separately. The input for Probabilistic neural network is arranged in such a way that all the vectors of a fault are bunched together (i.e. all vectors pertaining to a fault are placed adjacent to each other). Such a stack of one fault is followed by the stack of the next fault under consideration and so on. For seven faults the size of input is 36×245 and for twenty faults it is 36×700. Input vectors arranged for 20 faults are shown in Table 5.9. Fig 5.8 shows the typical plots all 35 vectors of a single fault. The target output is defined as 1 for first fault, 2 for second fault, 3 for third fault and so on. For seven faults it took 3-4 seconds to complete the training and for twenty faults it took 10-20 seconds to complete the training on a Pentium P-IV machine with 1.7 GHz. The performance of the Probabilistic neural network for classification of faults for training and testing for seven faults is given in Fig.5.9. Validation of the network is done with the testing data set of size 36×70. Performance of PNN for the case of twenty faults is given in Fig.5.10 for training and testing.

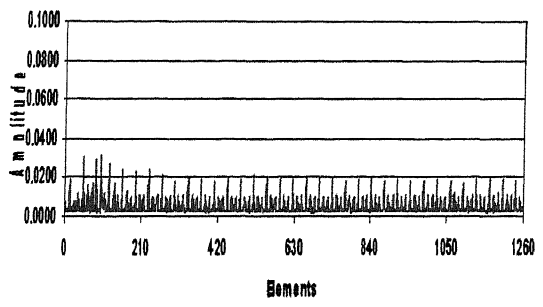
From Fig. 5.9 and Fig 5.10 it is clear that all faults except the No fault Case and Bearing Ball Spin defect Case do get identified properly. This phenomenon is observed whether the training is carried out for seven faults or twenty faults. Also, the mentioned faults do not get identified while training as well as validation.



(a) No Fault (b) Mass Unbalance

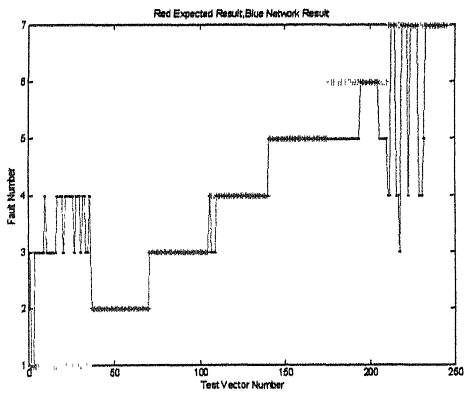
**Fig 5.8 Typical plots of all 35 vectors of a single fault.**

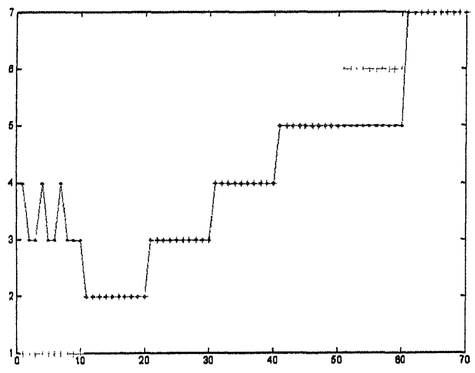




**(c ) Parallel Misalignment** **(d) Angular Misalignment**

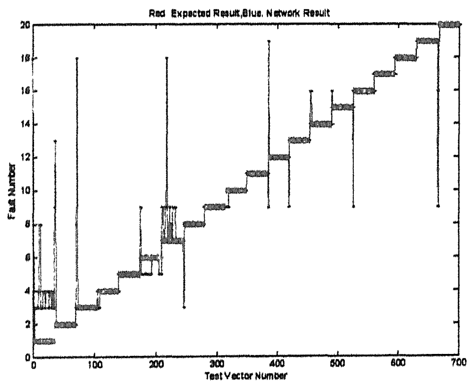
**Fig 5.8(Contd.) Typical plots of all 35 vectors of a single fault.**





**(a) Training** **(b) Testing**

**Fig 5.9 PNN Performance for seven Faults**



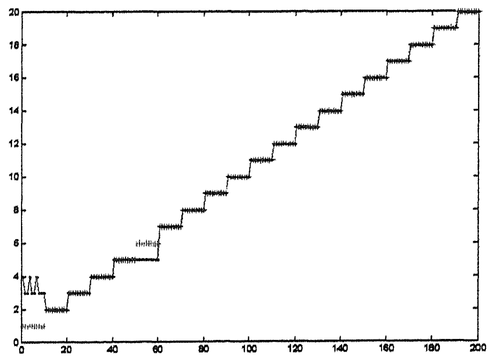


**Fig 5.10 PNN Performance for twenty faults**

## 5.5 Training with Self-Organizing Maps:

Back Propagation and Probabilistic Neural Network training is supervised training because in both the case the network requires the target data set for the corresponding training data set. But in Self Organizing Maps the training is unsupervised because they do not require any target data set for there training. The SOM ideally suited for classification problems since similar vector grouped into different cluster. The input for SOM is prepared by keeping all the vectors of single fault at one place. The training vectors are prepared after normalizing the input vectors of PNN in the range of 0.1 - 0.9. Typical data after normalizing is shown in Table 5.10. Thus the size for input data for seven faults is 36×245 and 36×700 for twenty faults.

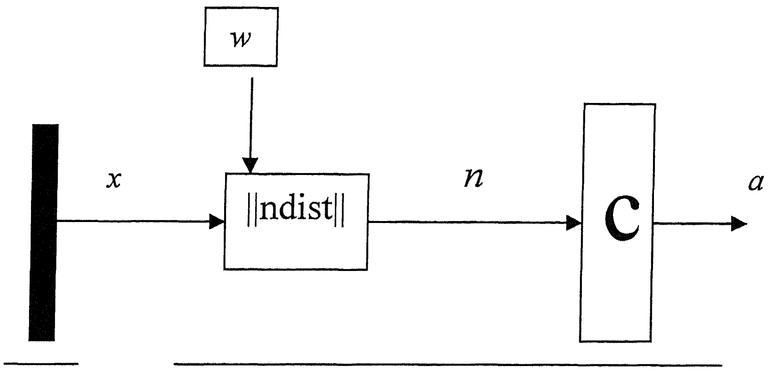
Training was carried out with 2-D and 3-D architecture. With 2-D architecture a 16-16 grid was chosen for diagnosis of seven faults and a 20-20 grid was chosen in the case with 20 faults. Training was carried out for 1000 epochs. With 3-D architecture a 15-15-4 grid was chosen for both seven fault case and twenty fault case. Training was carried out for 1000 epochs in the 3-D architecture also.

The learning process is explained step-by-step here for seven faults with a 2-D architecture of 16-16 grid.

Each fault is given a color code as shown below for some faults

| | | |
|---|---|---|
| 1. | No-Fault | ● |
| 2. | Mass Unbalance | ● |
| 3. | Parallel Misalignment | ● |
| 4. | Angular Misalignment | ○ |
| 5. | Inner Race Fault | ● |

The Self-Organising process involves mapping of these vectors on to the 16-16 grid shown earlier in Fig.3.8. Each input vector has a size (36 x 1). The training scheme is shown in Fig. 5.13, where the distance between the input vector and every row of the weight matrix (256 x 36) gets computed. In the next step, the minimum of the distance gets identified and declared winner. During the next epoch the process of Fig 5.11, is repeated after the weights have been modified by the Kohenen's learning rule, equation 3.25.



Input Self Organising Map Layer

$x$ = input vector of dimension R ×1.
$w$ = weight matrix (dimension R×S )for S neurons.
$n$ = distance (dimension S× 1) between the input vector and weight vector.
$a$ = output (size S×1; output I 1 for winning neuron and 0 for others)

**Fig. 5.11 Scheme in Training a Self Organising Map**

Learning in a self-organising map occurs for only one vector at a time. First the network identifies the winning neuron. Then the weights of the winning neuron, and the other neurons in it's neighborhood are moved closer to the input vector at each learning step using the self-organising map learning function. Because of this the network training is very slow and with the above-mentioned architectures the training took in excess of one day on a P-4 computer (1.7 GHz).

The training process, as it proceeds, is shown in Figs.5.15(a)-(f). Fig5.15(a) pertains to the neuron topology after just one epoch of learning. Fig5.15(b), (c),(d),(e) and (f) shows the topology after 10 epochs, 50 epochs, 100 epochs,500 epochs and 1000 epochs respectively. Comparing the grid before the initiation of learning (Fig.5.12) with that after 1 epoch (Fig. 5.13(a)), it can be seen that the neurons have started dispersing out of the centre. There is a lot of overlapping in specific locations and only those neurons, which were the last to map are visible. Others are hidden below them. Further dispersion takes place the process continues and and neurons of identical colour start assembly in different locations. The clustering consolidates as the number of epochs increase. Beyond a certain number of epochs, it is found that further consolidation of topology is not very significant. This is indicative of the convergence of the process. However, in the present study no attempt has been made to quantify the consolidation and convergence processes. During the present study training was carried out for 1000 epochs in all cases. Fig.5.14 shows training with 5 harmonics for seven faults and Fig.5.15 shows validation for the same. Fig.5.16 shows 7 fault training for seven faults and Fig 5.17 shows validation for the same for 12 harmonics. Fig.5.18 shows the training for 7 faults for 3 D architecture and Fig.5.19 shows validation. Training and validation for 2D and 3D for 20 faults with 12 harmonics is given from Fig.5.20,Fig.5.21;Fig.5.22 and Fig.5.23 respectively.

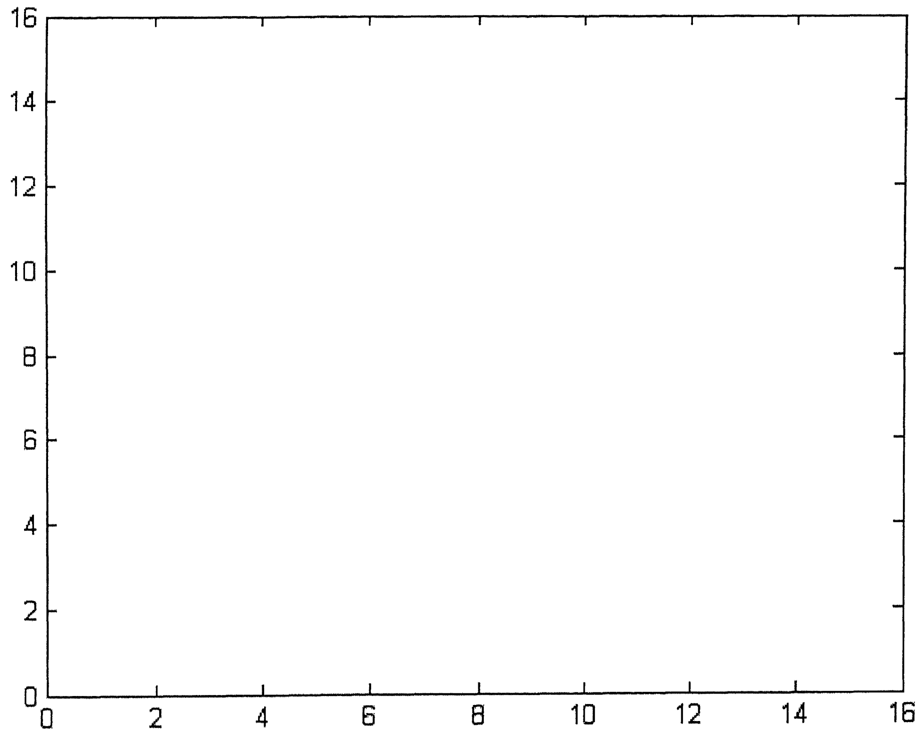
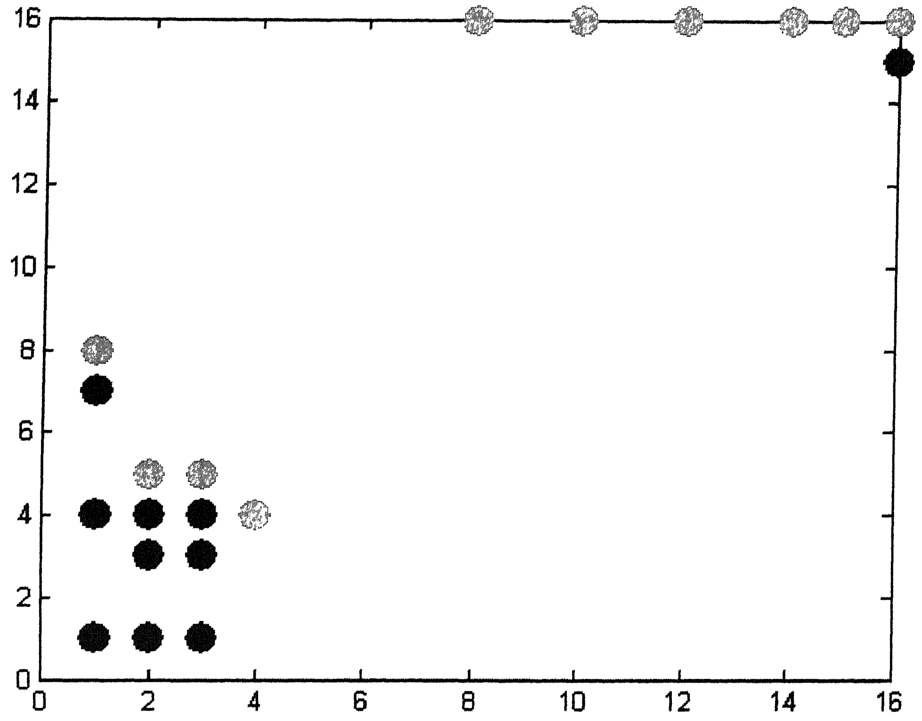
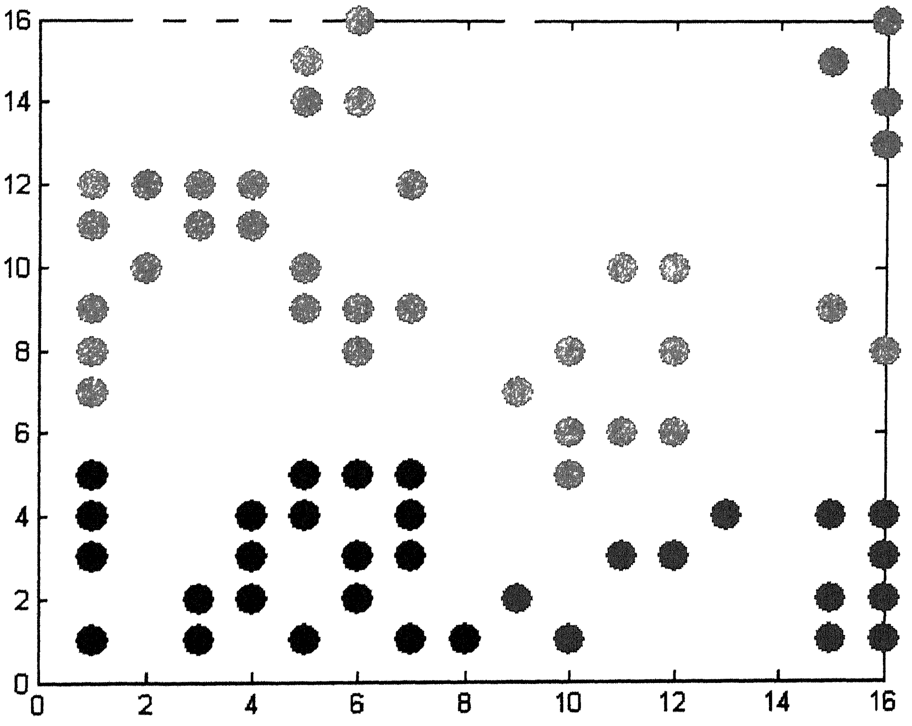


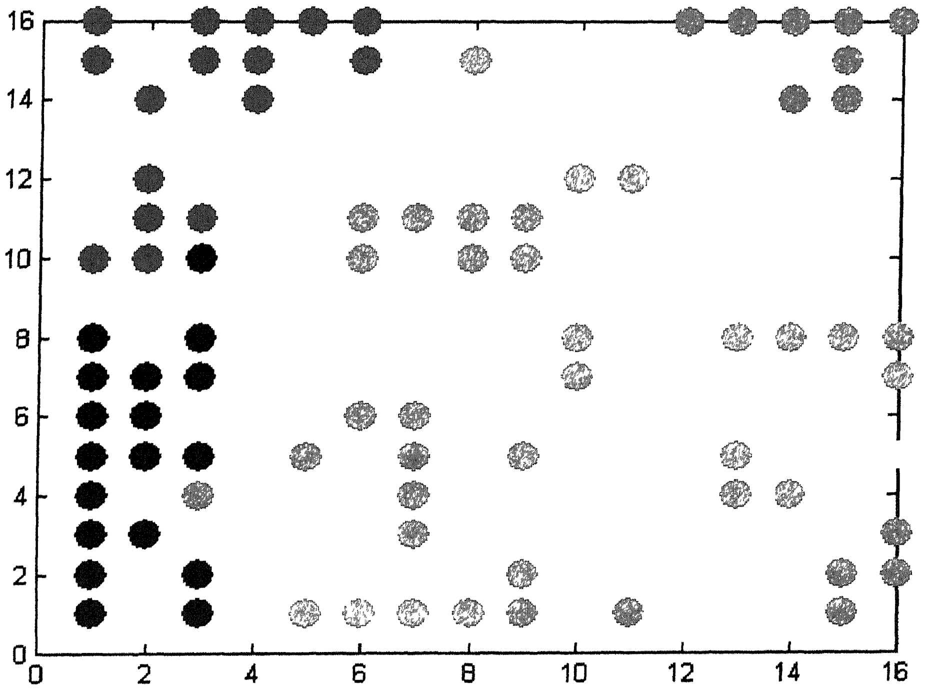
**Fig 5.12 All the vectors at the same place at the beginning of training.**



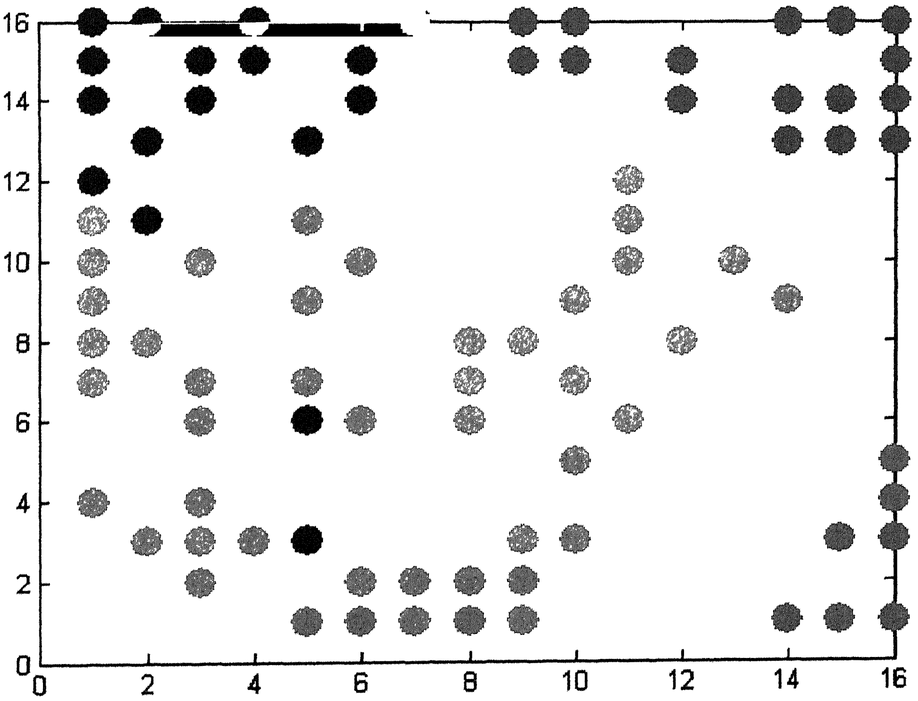
**(a) Neurons after 1 epoch of training**



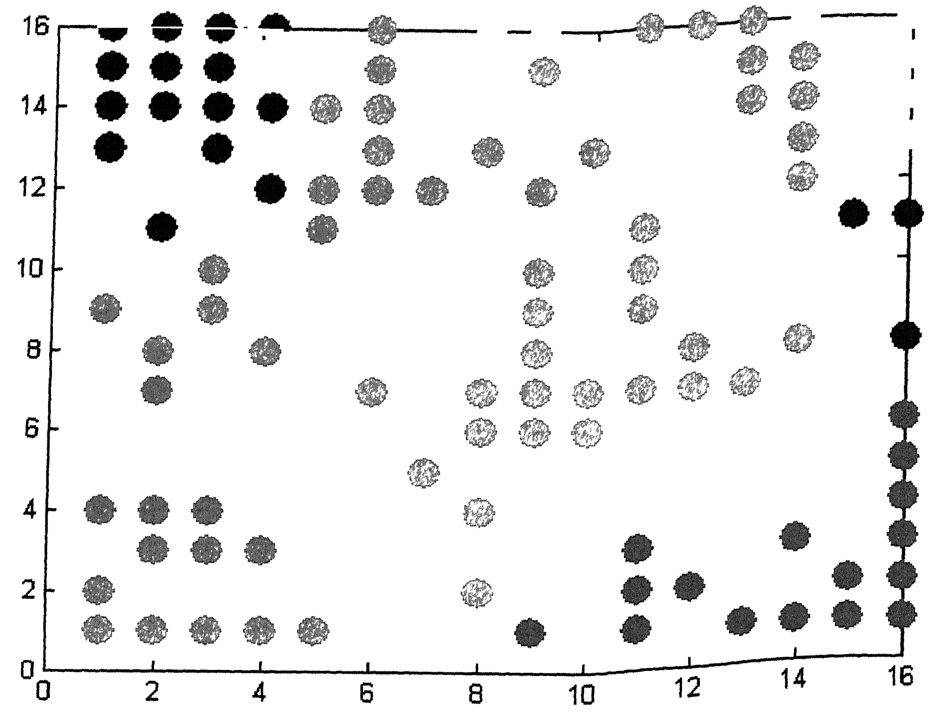
**(b) Neurons after 10 epochs of training**



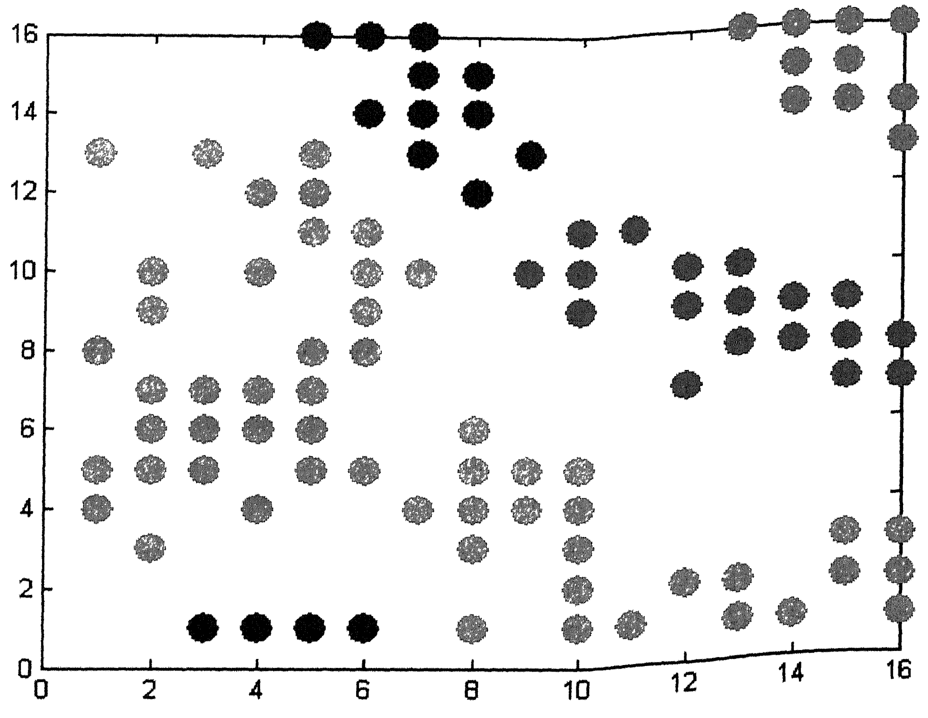
(c) Neurons after 50 epochs of training



(d) Neurons after 100 epochs of training



(e) Neurons after 500 epochs of training



(f) Neurons after 1000 epochs of training

**Fig 5.13 Convergence of SOM**

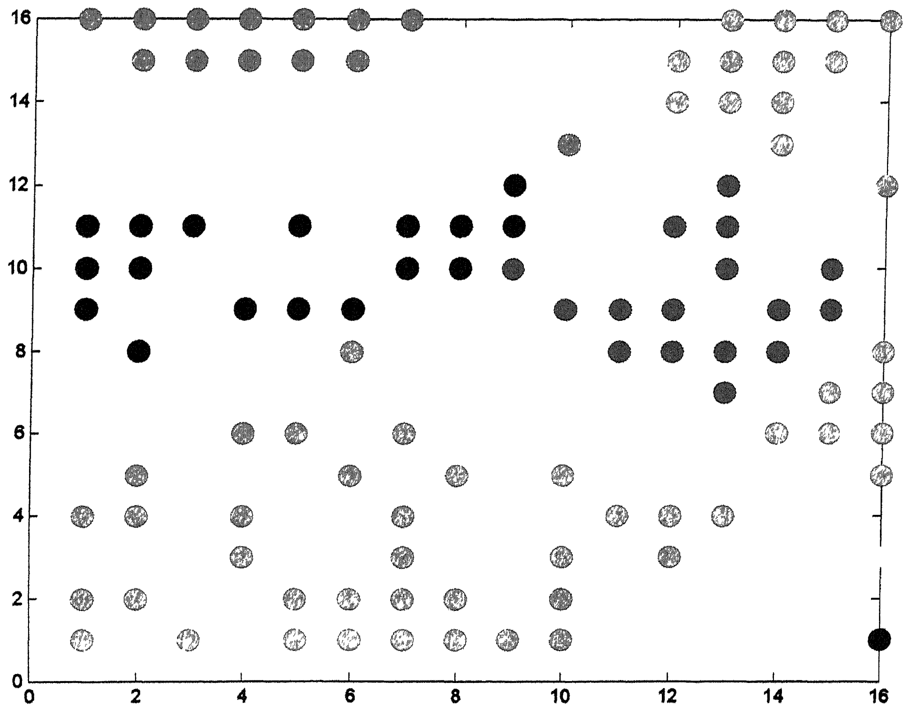


**Fig 5.14 Training with SOM architecture 16-16 for seven faults (5 harmonics)**

- ● No fault
- ● Mass Unbalance
- ● Parallel Misalignment
- ○ Angular Misalignment
- ● Bearing Inner race Fault
- ● Bearing Outer race Fault
- ● Bearing Ball spin

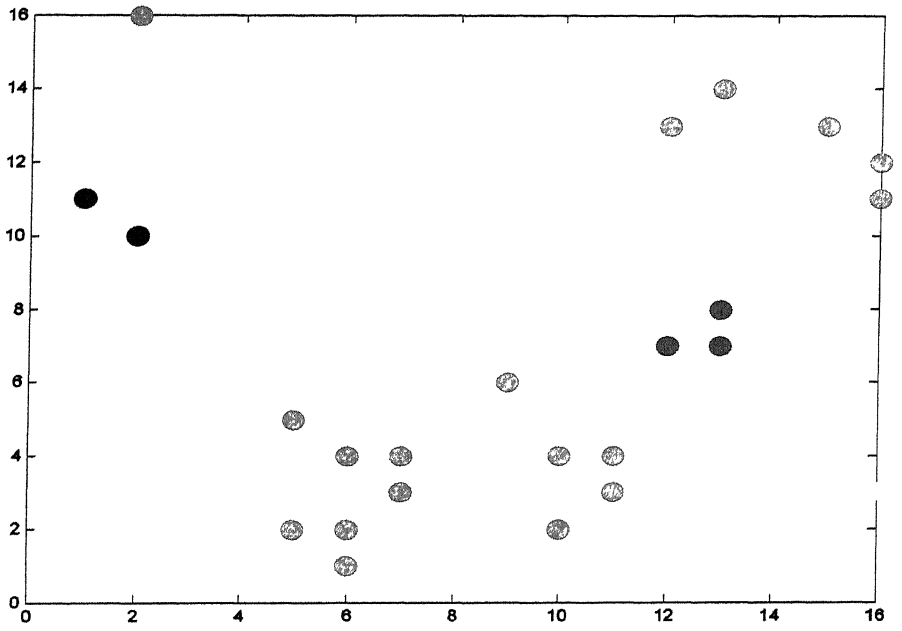


**Fig 5.15 Testing with SOM architecture 16-16 for seven faults.(5 harmonics)**

| | |
|---|---|
| ○ | **No fault** |
| ● | **Mass Unbalance** |
| ● | **Parallel Misalignment** |
| ○ | **Angular Misalignment** |
| ● | **Bearing Inner race Fault** |
| ● | **Bearing Outer race Fault** |
| ● | **Bearing Ball spin** |

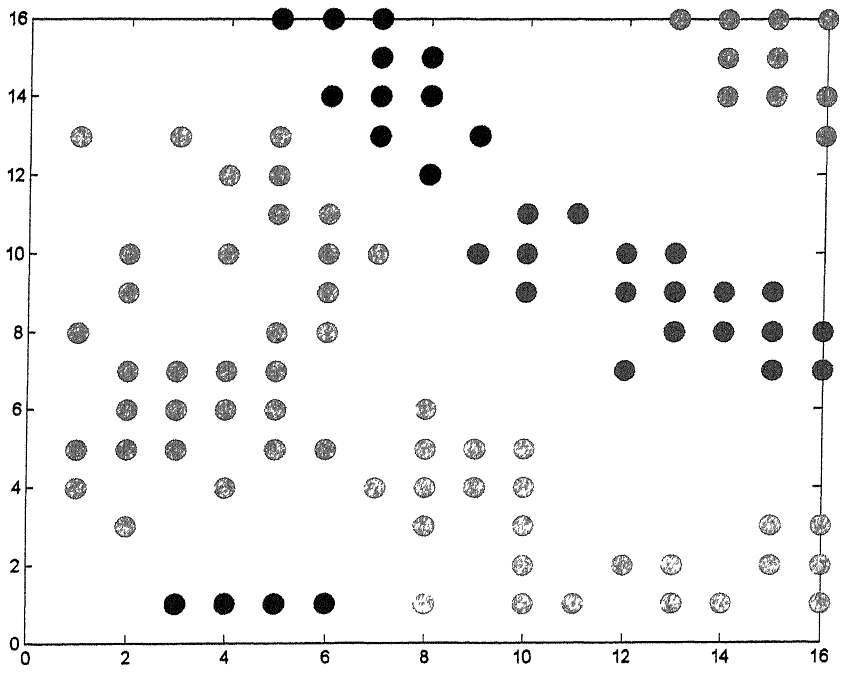


**Fig 5.16 Training with SOM architecture 16-16 for seven faults (12 harmonics)**

◍ **No fault**

◍ **Mass Unbalance**

◍ **Parallel Misalignment**

○ **Angular Misalignment**

● **Bearing Inner race Fault**

◍ **Bearing Outer race Fault**

● **Bearing Ball spin**

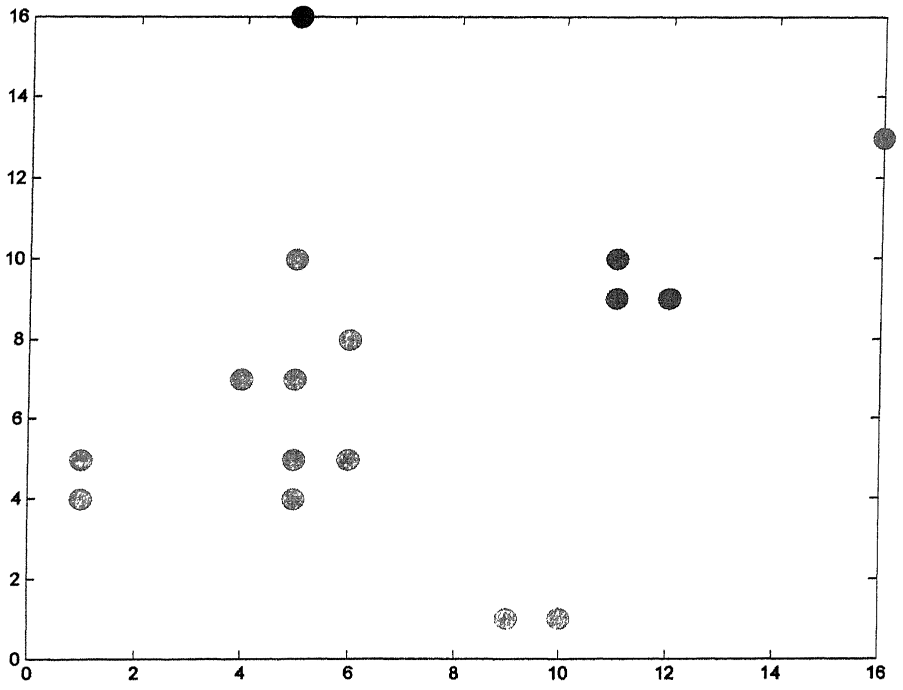


**Fig 5.17 Testing with SOM architecture 16-16 for seven faults (12 harmonics)(unnormalized)**

● **No fault**

● **Mass Unbalance**

● **Parallel Misalignment**

○ **Angular Misalignment**

● **Bearing Inner race Fault**

● **Bearing Outer race Fault**

● **Bearing Ball spin**

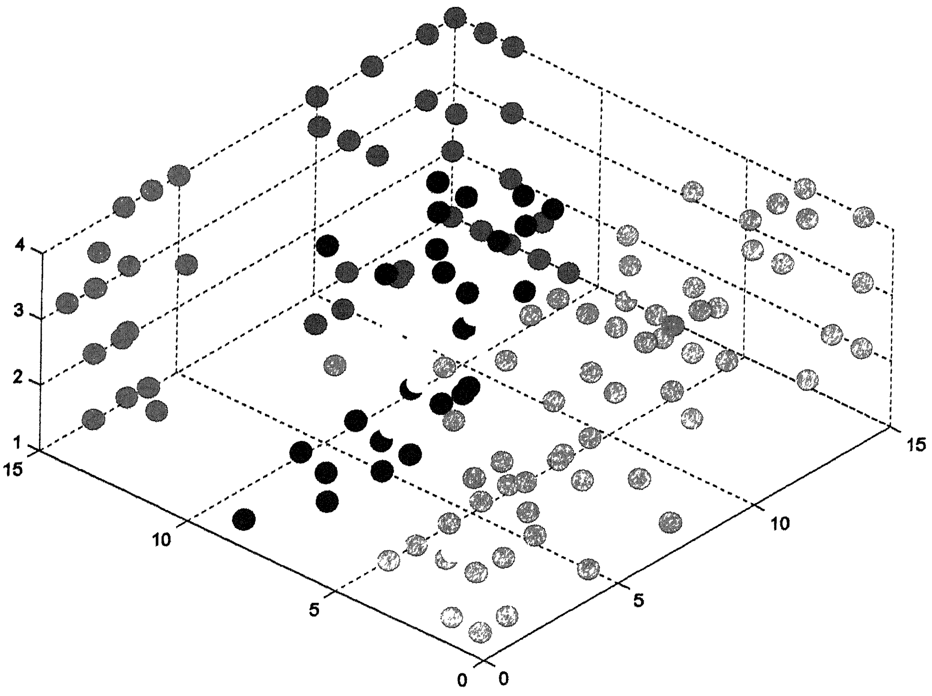


**Fig 5.18 Training with SOM architecture 15-15-4(normalized) for seven faults**

◍ **No fault**

● **Mass Unbalance**

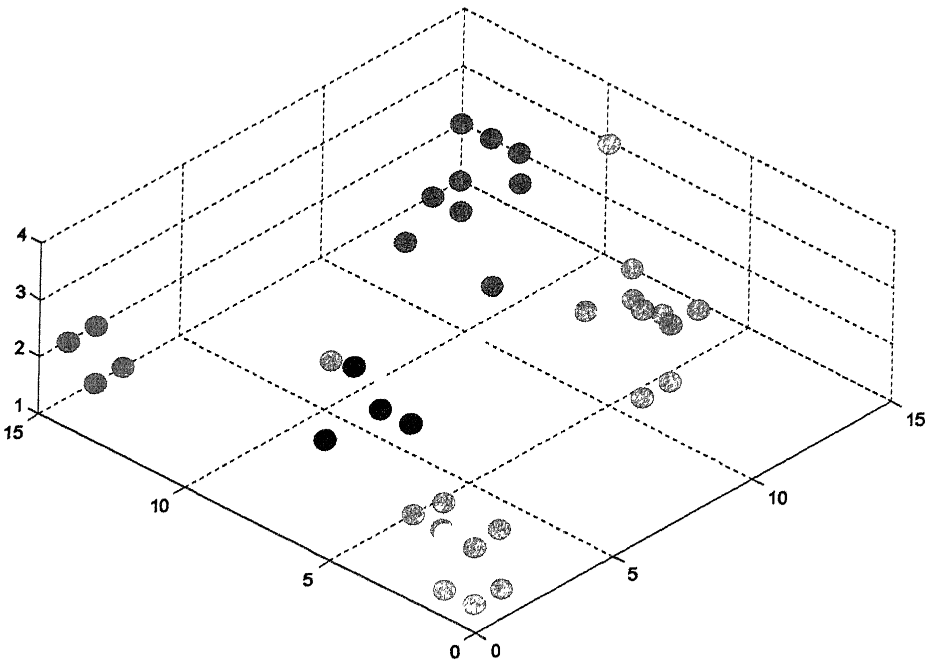
◍ **Parallel Misalignment**

○ **Angular Misalignment**

● **Bearing Inner race Fault**

◍ **Bearing Outer race Fault**

● **Bearing Ball spin**



**5.19 Testing with SOM architecture 15-15-4(normalized) for seven faults**

● **No fault**

● **Mass Unbalance**

● **Parallel Misalignment**

○ **Angular Misalignment**

● **Bearing Inner race Fault**

● **Bearing Outer race Fault**

● **Bearing Ball spin**

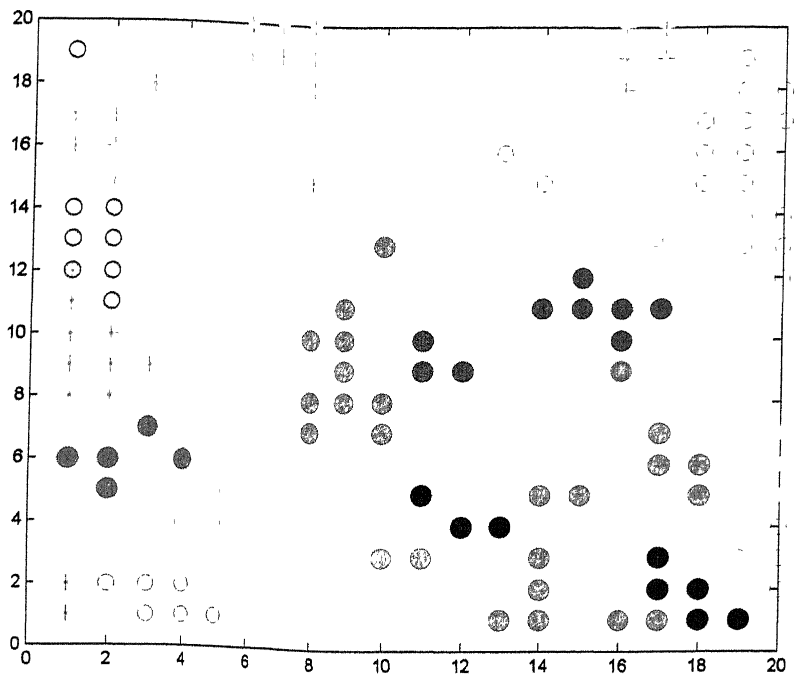


Fig 5.20 Training with SOM architecture 20-20(normalized) for twenty faults

| | | | |
|---|---|---|---|
| ◉ | No fault | | Reciprocating |
| ● | Mass Unbalance | ○ | Parallel Misalignment +Unbalance |
| ◉ | Parallel Misalignment | ○ | Angular Misalignment +Unbalance |
| ○ | Angular Misalignment | ○ | Inner race +Unbalance |
| ● | Bearing Inner race Fault | ☆ | Outer race +Unbalance |
| ◉ | Bearing Outer race Fault | ☆ | Ball Spin+ Unbalance |
| ● | Bearing Ball spin | ☆ | Ball Spin + gear Box |
| ○ | Cocked Rotor | | Cocked Rotor +Parallel Misalignment |
| ○ | Loose Belt | ☆ | Cocked Rotor + Unbalance |
| ○ | Tight Belt | ☆ | Reciprocating +Unbalance |

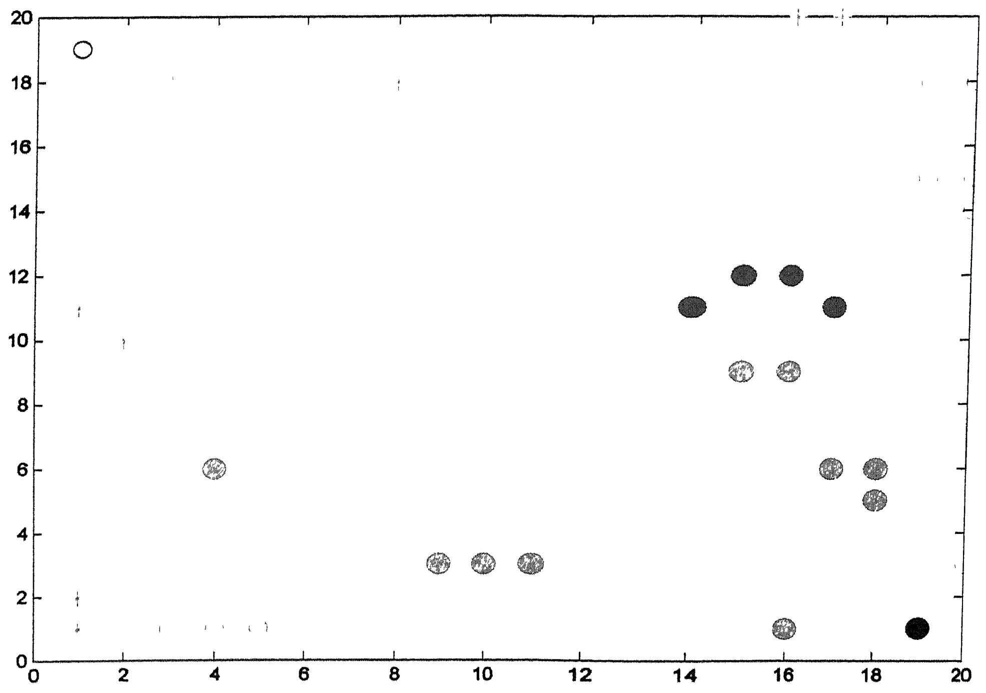


**Fig 5.21 Testing with SOM architecture 20-20(normalized) for twenty faults**

| | | | |
|---|---|---|---|
| ● | **No fault** | | **Reciprocating** |
| ● | **Mass Unbalance** | ○ | **Parallel Misalignment +Unbalance** |
| ● | **Parallel Misalignment** | ○ | **Angular Misalignment +Unbalance** |
| ○ | **Angular Misalignment** | ○ | **Inner race +Unbalance** |
| ● | **Bearing Inner race Fault** | ☆ | **Outer race +Unbalance** |
| ● | **Bearing Outer race Fault** | ☆ | **Ball Spin+ Unbalance** |
| ● | **Bearing Ball spin** | ☆ | **Ball Spin + gear Box** |
| ○ | **Cocked Rotor** | | **Cocked Rotor +Parallel Misalignment** |
| ○ | **Loose Belt** | ☆ | **Cocked Rotor + Unbalance** |
| ○ | **Tight Belt** | ☆ | **Reciprocating +Unbalance** |

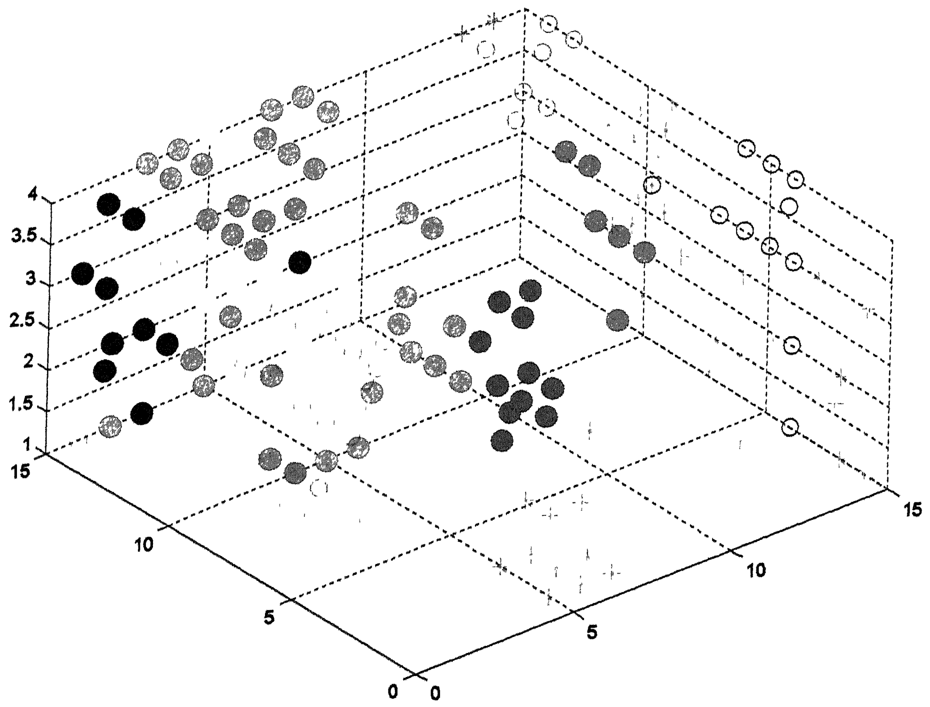


**Fig 5.22** **Training with SOM architecture 15-15-4(normalized) for twenty faults**

| | | | |
|---|---|---|---|
| ● | **No fault** | | **Reciprocating** |
| ● | **Mass Unbalance** | ○ | **Parallel Misalignment +Unbalance** |
| ● | **Parallel Misalignment** | ○ | **Angular Misalignment +Unbalance** |
| ○ | **Angular Misalignment** | ○ | **Inner race +Unbalance** |
| ● | **Bearing Inner race Fault** | ☆ | **Outer race +Unbalance** |
| ● | **Bearing Outer race Fault** | ☆ | **Ball Spin+ Unbalance** |
| ● | **Bearing Ball spin** | ☆ | **Ball Spin + gear Box** |
| ○ | **Cocked Rotor** | | **Cocked Rotor +Parallel Misalignment** |
| ○ | **Loose Belt** | ☆ | **Cocked Rotor + Unbalance** |
| ○ | **Tight Belt** | ☆ | **Reciprocating +Unbalance** |

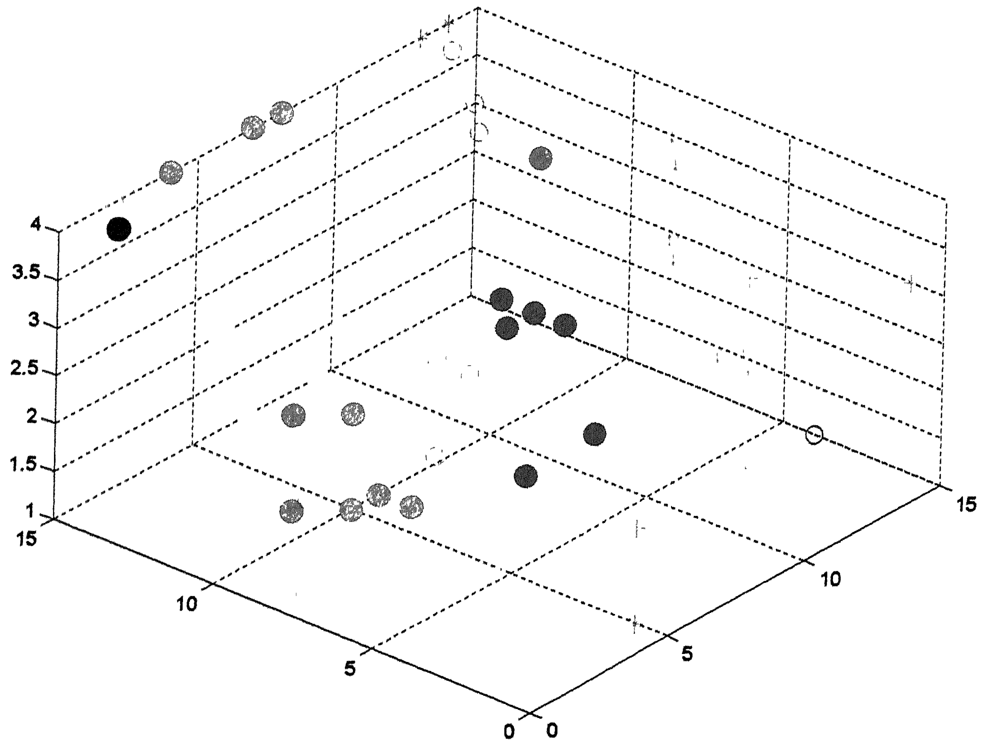


**Fig 5.23** **Testing with SOM architecture 15-15-4(normalized) for twenty faults**

| | | | |
|---|---|---|---|
| ● | **No fault** | | **Reciprocating** |
| ● | **Mass Unbalance** | ○ | **Parallel Misalignment +Unbalance** |
| ● | **Parallel Misalignment** | ○ | **Angular Misalignment +Unbalance** |
| ○ | **Angular Misalignment** | ○ | **Inner race +Unbalance** |
| ● | **Bearing Inner race Fault** | ☆ | **Outer race +Unbalance** |
| ● | **Bearing Outer race Fault** | ☆ | **Ball Spin+ Unbalance** |
| ● | **Bearing Ball spin** | ☆ | **Ball Spin + gear Box** |
| ○ | **Cocked Rotor** | | **Cocked Rotor +Parallel Misalignment** |
| ○ | **Loose Belt** | ☆ | **Cocked Rotor + Unbalance** |
| ○ | **Tight Belt** | ☆ | **Reciprocating +Unbalance** |

Table 5.11 Details of various Neurons and their overlapping

| Neuron No. | Fault No. | Same Fault Count | Other Fault Count | Total Count |
|---|---|---|---|---|
| 2 | 1 | 1 | 1 | 2 |
| 7 | 1 | 2 | 0 | 2 |
| 9 | 1 | 3 | 0 | 3 |
| 14 | 1 | 3 | 0 | 3 |
| 17 | 1 | 1 | 0 | 1 |
| 18 | 1 | 8 | 0 | 8 |
| 21 | 1 | 1 | 0 | 1 |
| 36 | 2 | 0 | 1 | 1 |
| 37 | 2 | 1 | 0 | 1 |
| 39 | 2 | 5 | 0 | 5 |
| 45 | 2 | 2 | 0 | 2 |
| 46 | 2 | 5 | 0 | 5 |
| 55 | 2 | 2 | 0 | 2 |
| 56 | 2 | 1 | 0 | 1 |
| 59 | 2 | 4 | 0 | 4 |
| 66 | 2 | 4 | 0 | 4 |
| 74 | 3 | 1 | 0 | 1 |
| 76 | 3 | 1 | 0 | 1 |
| 78 | 3 | 2 | 0 | 2 |
| 81 | 3 | 4 | 0 | 4 |
| 82 | 3 | 3 | 0 | 3 |
| 87 | 3 | 2 | 0 | 2 |
| 88 | 3 | 4 | 0 | 4 |
| 97 | 3 | 1 | 0 | 1 |
| 106 | 4 | 1 | 0 | 1 |
| 107 | 4 | 1 | 0 | 1 |
| 110 | 4 | 1 | 0 | 1 |
| 113 | 4 | 8 | 0 | 8 |
| 115 | 4 | 9 | 0 | 9 |
| 117 | 4 | 7 | 0 | 7 |
| 141 | 5 | 2 | 0 | 2 |
| 146 | 5 | 1 | 0 | 1 |
| 149 | 5 | 1 | 0 | 1 |
| 150 | 5 | 1 | 0 | 1 |
| 153 | 5 | 2 | 0 | 2 |
| 154 | 5 | 2 | 0 | 2 |
| 159 | 5 | 1 | 0 | 1 |
| 163 | 5 | 1 | 0 | 1 |
| 166 | 5 | 1 | 0 | 1 |
| 167 | 5 | 1 | 0 | 1 |
| 170 | 5 | 4 | 0 | 4 |
| 179 | 6 | 1 | 0 | 1 |
| 182 | 6 | 1 | 0 | 1 |
| 183 | 6 | 4 | 0 | 4 |
| 198 | 6 | 3 | 0 | 3 |
| 206 | 6 | 3 | 0 | 3 |
| 212 | 7 | 4 | 0 | 4 |
| 213 | 7 | 2 | 0 | 2 |
| 218 | 7 | 1 | 0 | 1 |
| 219 | 7 | 1 | 0 | 1 |
| 220 | 7 | 1 | 0 | 1 |
| 225 | 7 | 1 | 0 | 1 |
| 229 | 7 | 1 | 0 | 1 |
| 234 | 7 | 1 | 0 | 1 |
| 235 | 7 | 2 | 0 | 2 |
| 238 | 7 | 6 | 0 | 6 |

**Table 5.12 7- fault training with architecture 16-16(5 harmonics)**

| Fault Name | Total No. of Neurons | self overlapped | Incoming | Outgoing | Interchange | no overlapping |
|---|---|---|---|---|---|---|
| No Fault | 35 | 24 | 2 | 0 | 2 | 9 |
| Mass Unbalance | 35 | 32 | 0 | 0 | 0 | 3 |
| Parallel Misalignment | 35 | 26 | 0 | 1 | 1 | 9 |
| Angular Misalignment | 35 | 28 | 1 | 1 | 1 | 6 |
| Bearing Inner Race Fault | 35 | 29 | 0 | 0 | 0 | 6 |
| Bearing Outer Race Fault | 35 | 22 | 0 | 0 | 0 | 13 |
| Bearing Ball Spin | 35 | 25 | 0 | 1 | 1 | 10 |

**Table 5.13 7-fault testing with architecture 16-16(5 harmonics)**

| Fault Name | Total | self overlapped | Incomming | Outgoing | interchange | no overlapping |
|---|---|---|---|---|---|---|
| No Fault | 10 | 6 | 0 | 0 | 0 | 4 |
| Mass Unbalance | 10 | 10 | 0 | 0 | 0 | 0 |
| Parallel Misalignment | 10 | 7 | 0 | 0 | 0 | 3 |
| Angular Misalignment | 10 | 9 | 0 | 0 | 0 | 1 |
| Bearing Inner Race Fault | 10 | 8 | 0 | 0 | 0 | 2 |
| Bearing Outer Race Fault | 10 | 7 | 0 | 0 | 0 | 3 |
| Bearing Ball Spin | 10 | 10 | 0 | 0 | 0 | 0 |

**Table 5.14 7 -fault training with architecture 16-16(12 harmonics)**

| Fault Name | Total | self overlapped | Incomming | Outgoing | interchange | no overlapping |
|---|---|---|---|---|---|---|
| No Fault | 35 | 26 | 5 | 0 | 5 | 9 |
| Mass Unbalance | 35 | 32 | 9 | 4 | 13 | 2 |
| Parallel Misalignment | 35 | 26 | 1 | 7 | 8 | 9 |
| Angular Misalignment | 35 | 33 | 4 | 4 | 8 | 2 |
| Bearing Inner Race Fault | 35 | 28 | 3 | 4 | 7 | 7 |
| Bearing Outer Race Fault | 35 | 17 | 5 | 3 | 8 | 18 |
| Bearing Ball Spin | 35 | 30 | 0 | 5 | 5 | 5 |

**Table 5.15 7- fault testing with architecture 16-16(12 harmonics)**

| Fault Name | Total | self overlapped | Incomming | Outgoing | interchange | no overlapping |
|---|---|---|---|---|---|---|
| No Fault | 10 | 9 | 0 | 0 | 0 | 1 |
| Mass Unbalance | 10 | 10 | 0 | 0 | 0 | 0 |
| Parallel Misalignment | 10 | 8 | 0 | 0 | 0 | 2 |
| Angular Misalignment | 10 | 10 | 0 | 0 | 0 | 0 |
| Bearing Inner Race Fault | 10 | 9 | 0 | 0 | 0 | 1 |
| Bearing Outer Race Fault | 10 | 10 | 0 | 0 | 0 | 0 |
| Bearing Ball Spin | 10 | 10 | 0 | 0 | 0 | 0 |

**Table 5.16 7- fault training with architecture 15-15-4**

| Fault Name | Total | self overlapped | Incomming | Outgoing | interchange | no overlapping |
|---|---|---|---|---|---|---|
| No Fault | 35 | 28 | 3 | 0 | 3 | 6 |
| Mass Unbalance | 35 | 28 | 2 | 0 | 2 | 6 |
| Parallel Misalignment | 35 | 15 | 1 | 3 | 4 | 19 |
| Angular Misalignment | 35 | 25 | 0 | 3 | 3 | 10 |
| Bearing Inner Race Fault | 35 | 16 | 0 | 0 | 0 | 19 |
| Bearing Outer Race Fault | 35 | 23 | 0 | 0 | 0 | 12 |
| Bearing Ball Spin | 35 | 14 | 0 | 0 | 0 | 21 |

**Table5.17 7 -fault testing with architecture 15-15-4**

| Fault Name | Total | self overlapped | Incomming | Outgoing | interchange | no overlapping |
|---|---|---|---|---|---|---|
| No Fault | 10 | 2 | 0 | 0 | 0 | 8 |
| Mass Unbalance | 10 | 8 | 0 | 0 | 0 | 2 |
| Parallel Misalignment | 10 | 7 | 0 | 0 | 0 | 3 |
| Angular Misalignment | 10 | 5 | 0 | 0 | 0 | 5 |
| Bearing Inner Race Fault | 10 | 2 | 0 | 0 | 0 | 8 |
| Bearing Outer Race Fault | 10 | 9 | 0 | 0 | 0 | 1 |
| Bearing Ball Spin | 10 | 9 | 0 | 0 | 0 | 1 |

**Table5.18 20-fault training architecture 15-15-4**

| Fault Name | Total | self overlapped | Incomming | Outgoing | interchange | no overlapping |
|---|---|---|---|---|---|---|
| No Fault | 35 | 29 | 9 | 0 | 9 | 4 |
| Mass Unbalance | 35 | 33 | 0 | 0 | 0 | 2 |
| Parallel Misalignment | 35 | 29 | 1 | 2 | 3 | 6 |
| Angular Misalignment | 35 | 27 | 1 | 4 | 5 | 8 |
| Bearing Inner Race Fault | 35 | 31 | 0 | 0 | 0 | 4 |
| Bearing Outer Race  Fault | 35 | 32 | 0 | 0 | 0 | 3 |
| Bearing Ball Spin | 35 | 29 | 1 | 1 | 2 | 6 |
| Cocked Rotor | 35 | 28 | 0 | 4 | 4 | 7 |
| Loose Belt | 35 | 31 | 0 | 0 | 0 | 4 |
| Tight Belt | 35 | 30 | 0 | 0 | 0 | 5 |
| Reciprocating | 35 | 29 | 0 | 0 | 0 | 6 |
| Parallel Misalignment+Unbalance | 35 | 32 | 0 | 0 | 0 | 3 |
| Angular Misalignment+Unbalance | 35 | 33 | 0 | 0 | 0 | 2 |
| Inner race+Unbalance | 35 | 30 | 5 | 0 | 5 | 4 |
| Outer race+Unbalance | 35 | 32 | 0 | 0 | 0 | 3 |
| Ball Spin +Unbalance | 35 | 25 | 0 | 5 | 5 | 10 |
| Ball Spin +Gear Box | 35 | 31 | 0 | 0 | 0 | 4 |
| Cocked Rotor +Parallel Misalignment | 35 | 34 | 0 | 0 | 0 | 1 |
| Cocked Rotor+Unbalance | 35 | 34 | 0 | 0 | 0 | 2 |
| Reciprocating+Unbalance | 35 | 32 | 0 | 1 | 1 | 3 |

**Table 5.19 20-fault testing with architecture 15-15-4**

| Fault Name | Total | self overlapped | Incomming | Outgoing | interchange | no overlapping |
|---|---|---|---|---|---|---|
| No Fault | 10 | 10 | 9 | 0 | 9 | 0 |
| Mass Unbalance | 10 | 10 | 0 | 0 | 0 | 0 |
| Parallel Misalignment | 10 | 10 | 0 | 0 | 0 | 0 |
| Angular Misalignment | 10 | 7 | 0 | 0 | 0 | 3 |
| Bearing Inner Race Fault | 10 | 8 | 0 | 0 | 0 | 2 |
| Bearing Outer Race Fault | 10 | 9 | 0 | 0 | 0 | 1 |
| Bearing Ball Spin | 10 | 10 | 0 | 0 | 0 | 0 |
| Cocked Rotor | 10 | 9 | 0 | 0 | 0 | 1 |
| Loose Belt | 10 | 10 | 0 | 0 | 0 | 0 |
| Tight Belt | 10 | 9 | 0 | 0 | 0 | 1 |
| Reciprocating | 10 | 10 | 0 | 0 | 0 | 1 |
| Parallel Misalignment+Unbalance | 10 | 9 | 0 | 0 | 0 | 3 |
| Angular Misalignment+Unbalance | 10 | 7 | 0 | 0 | 0 | 0 |
| Inner race+Unbalance | 10 | 10 | 0 | 0 | 0 | 0 |
| Outer race+Unbalance | 10 | 10 | 0 | 0 | 0 | 2 |
| Ball Spin +Unbalance | 10 | 8 | 0 | 0 | 0 | 0 |
| Ball Spin +Gear Box | 10 | 10 | 0 | 0 | 0 | 0 |
| Cocked Rotor +Parallel Misalignment | 10 | 10 | 0 | 0 | 0 | 0 |
| Cocked Rotor+Unbalance | 10 | 10 | 0 | 0 | 0 | 0 |
| Reciprocating+Unbalance | 10 | 8 | 0 | 0 | 0 | 2 |

**Table5.20 20-Fault training with architecture 20-20**

| Fault Name | Total | self overlapped | Incomming | Outgoing | interchange | no overlapping |
|---|---|---|---|---|---|---|
| No Fault | 35 | 31 | 9 | 0 | 9 | 2 |
| Mass Unbalance | 35 | 32 | 2 | 0 | 2 | 2 |
| Parallel Misalignment | 35 | 29 | 3 | 3 | 6 | 5 |
| Angular Misalignment | 35 | 28 | 0 | 5 | 0 | 7 |
| Bearing Inner Race Fault | 35 | 34 | 0 | 0 | 0 | 1 |
| Bearing Outer Race Fault | 35 | 33 | 0 | 0 | 0 | 2 |
| Bearing Ball Spin | 35 | 32 | 1 | 0 | 1 | 2 |
| Cocked Rotor | 35 | 31 | 0 | 3 | 3 | 4 |
| Loose Belt | 35 | 34 | 3 | 0 | 3 | 1 |
| Tight Belt | 35 | 27 | 0 | 3 | 3 | 8 |
| Reciprocating | 35 | 33 | 0 | 0 | 0 | 2 |
| Parallel Misalignment+Unbalance | 35 | 34 | 0 | 1 | 1 | 1 |
| Angular Misalignment+Unbalance | 35 | 34 | 0 | 1 | 1 | 1 |
| Inner race+Unbalance | 35 | 33 | 4 | 0 | 4 | 1 |
| Outer race+Unbalance | 35 | 33 | 0 | 0 | 0 | 2 |
| Ball Spin +Unbalance | 35 | 28 | 0 | 4 | 4 | 7 |
| Ball Spin +Gear Box | 35 | 35 | 0 | 0 | 0 | 0 |
| Cocked Rotor +Parallel Misalignment | 35 | 34 | 0 | 1 | 1 | 1 |
| Cocked Rotor+Unbalance | 35 | 34 | 0 | 0 | 0 | 1 |
| Reciprocating+Unbalance | 35 | 32 | 0 | 1 | 1 | 3 |

**Table 5.21 20-Fault testing with architecture 20-20**

| Fault Name | Total | self overlapped | Incomming | Outgoing | interchange | no overlapping |
|---|---|---|---|---|---|---|
| No Fault | 10 | 8 | 0 | 0 | 0 | 2 |
| Mass Unbalance | 10 | 10 | 0 | 0 | 0 | 0 |
| Parallel Misalignment | 10 | 9 | 0 | 0 | 0 | 1 |
| Angular Misalignment | 10 | 9 | 0 | 0 | 0 | 1 |
| Bearing Inner Race Fault | 10 | 9 | 0 | 0 | 0 | 1 |
| Bearing Outer Race Fault | 10 | 10 | 0 | 0 | 0 | 0 |
| Bearing Ball Spin | 10 | 10 | 0 | 0 | 0 | 0 |
| Cocked Rotor | 10 | 9 | 0 | 0 | 0 | 1 |
| Loose Belt | 10 | 8 | 0 | 0 | 0 | 2 |
| Tight Belt | 10 | 10 | 0 | 0 | 0 | 0 |
| Reciprocating | 10 | 9 | 0 | 0 | 0 | 1 |
| Parallel Misalignment+Unbalance | 10 | 9 | 0 | 0 | 0 | 1 |
| Angular Misalignment+Unbalance | 10 | 9 | 0 | 0 | 0 | 1 |
| Inner race+Unbalance | 10 | 10 | 0 | 0 | 0 | 0 |
| Outer race+Unbalance | 10 | 10 | 0 | 0 | 0 | 0 |
| Ball Spin +Unbalance | 10 | 9 | 0 | 0 | 0 | 1 |
| Ball Spin +Gear Box | 10 | 9 | 0 | 0 | 0 | 1 |
| Cocked Rotor +Parallel Misalignment | 10 | 10 | 0 | 0 | 0 | 0 |
| Cocked Rotor+Unbalance | 10 | 9 | 0 | 0 | 0 | 1 |
| Reciprocating+Unbalance | 10 | 10 | 0 | 0 | 0 | 0 |

## 5.6 Remarks

It is evident from Figs.5.14-5.22 that for both 2-D mapping and 3-D mapping, input neurons from the same fault get mapped together in clusters. Each fault occupies a specific zone on the grid. Table 5.11 shows the details of neurons and their overlapping on self group and on other groups. The results of these figures are given in tabular form in Tables 5.12-5.21. The Self-Organising activity of neurons needs to be understood simultaneously from the figures and tables, since the number of neurons visible on the grid is not the same as the total number of neurons. The tables give the total number of mapped neurons and their distribution on the grid, in terms of (i) those which overlap on neurons of their own kind, (ii) those which have overlap with neurons of other faults and (iii) those with no overlap of any kind. The number of neurons actually visible on the grid can be readily obtained as( total numbers of neurons (35 in case of training and 10 in case of validation)-total overlapping)(Table 5.11). It is neurons in categories (i) and (iii) that help in defining the fault zones. Those causing overlap lie in ambiguous zones, which do not allow fault diagnosis.

# CHAPTER 6

# CONCLUSIONS AND SCOPE FOR FUTURE WORK

The objective of the present work was to develop an automated system for diagnosis of rotor faults. The diagnosis procedure is based on Neural Network methodology employing concepts of Self-Organising Maps. The diagnosis algorithm has been integrated with the data-acquisition hardware and software. Additionally, features for audio and video warning alarm in case of fault development and capability for easy and automated data storage and retrieval has been developed.

The two areas of activity during the present study were (i) development of Virtual Instrumentation for a variety of data acquisition and signal processing functions from multiple sensors on a rotating machine and (ii) development of neural network based on self-organising maps. Activities in both areas were ongoing and specific contribution made through the present work can be listed as – (a) Development of Audio and Video Warning System within the VI; (b) Development of Data Storage with date/time identification; (c) Refinement of the FFT algorithm; (d) integration of the Neural Network MATLAB code in LabVIEW.

The Neural Network code based on Self-Organising Maps is found to provide better visual information in comparison to those based on Back-Propagation and Probabilistic concepts. While the algorithm for Self-Organising Maps is simple in comparison to the other two, training with Self-Organising maps is slow. Probabilistic Neural Network training is the fastest since it is a single-pass activity. In Back-Propagation schemes the training time depends on the level of required convergence limit. It is difficult to quantitatively define convergence levels in Self-Organising Maps.

In the present study an attempt has been made to study the overlapping patterns of neurons on the Self-Organising Map grid. It is found that for most rotor faults the neurons map on separate regions. The exercise has been carried out for seven number of faults in one case and twenty faults in another case. The performance of Self Organising Maps has been found to be good. The number of neurons overlapping with those from other faults can be taken as the error involved in the fault identification scheme.

Further effort is required to quantitatively define fault nuclei and their influence zones on 2-D and 3-D grids of Self-Organising. It is also necessary to study convergence aspects of these maps quantitatively. A study is also required on the optimum size of the input vector and the appropriate number of harmonics to be extracted from FFT signal of the rotor.

# REFERENCES

1. Bose, N.K., and Liang, P., *Neural Network Foundations, With Graphs, Algorithms, and Applications*, Tata McGraw-Hill, New Delhi, 1996.
2. Childs D., *Turbo-machinery Rotordynamics*, John Wiley & Sons, 1993.
3. Dimentberg M.F., *Statistical Dynamics of Nonlinear and Time-Varying Systems*, Applied Science Publishers, 1983.
4. Eisenmann R.C. Sr., Eisenmann R.C. Jr., Machinery Malfunction Diagnosis and Correction, Prentice-Hall PTR, New Jersey, 1998.
5. Elkordy, M., Chang, K.C., and Lee, G.C., "Application of Neural Networks in Vibrational Signatures Analysis", Journal of Engineering Mechanics, ASCE, Vol. 120, No. 2, pp 250-265, February, 1994.
6. Hassoun, S., Fundamentals of Artificial Neural Networks, Prentice-Hall of India Private Ltd, 1999.
7. Mayes I.W., "Use of neural network for on-line vibration monitoring", Proceedings of Institute of Mechanical Engineers, Vol. 208, pp 267-2741994.
8. McCormick A.C., Nandi A.K., "Real time classification of rotating shaft loading conditions using artificial neural networks", IEEE Transactions on Neural Networks, Vol. 8, No. 3, pp 748-757, May 1997.
9. Norton, M.P., Fundamentals of Noise and Vibration analysis for Engineers, Cambridge University Press, Cambridge, 1989.
10. Padhy, D.K., " Fault identification in Rotating Machinery using Neural Networks and Virtual Instrumentation", M. Tech thesis, Jan 2001, IIT Kanpur
11. Rao J.S., Rotor Dynamics, Third edition, New Age International (P) Limited, New Delhi, 1996.
12. Sohre, J.S., "Turbo machinery Problems and Their Correction", Standardization and Condition Monitoring Workshop, Chapter 7, Houston, 1991.
13. Vyas N.S., Jain N., Pandey S., "Fault identification in rotating machinery using neural networks", accepted for publication in the Journal of Vibration Institute of India, 2000.
14. Zurada, M.J., Introduction to Artificial Neural Systems, Jaico Publishing House, Delhi, 1999.